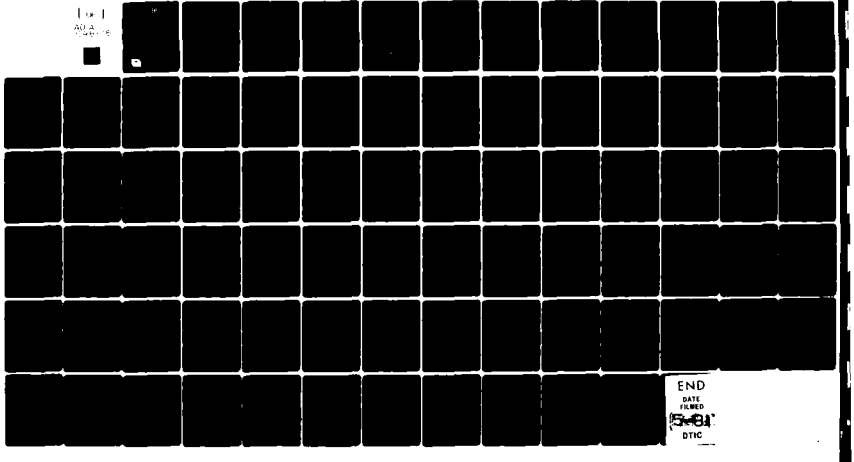
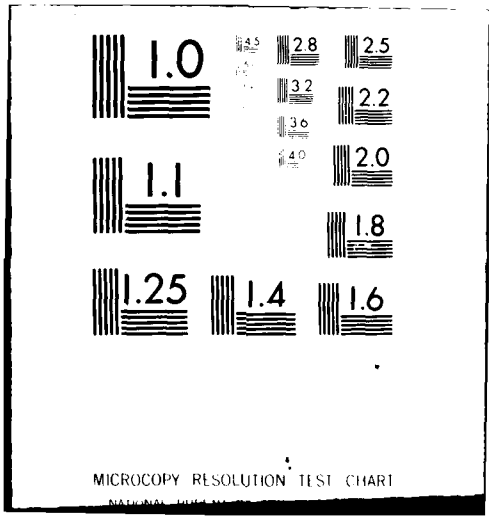
AD-A098 176 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA SCIENCE A--ETC F/6 9/5
VERY LARGE SCALE INTEGRATED CIRCUITS FOR MILITARY SYSTEMS.(U)
JAN 81 G W PRESTON MDA903-79-C-0202

UNCLASSIFIED IDA-P-1538 IDA/HQ-80-22941 NL



END
DATE FILMED
5-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART

(2) LEVEL III

IDA PAPER P-1538

VERY LARGE SCALE INTEGRATED CIRCUITS FOR MILITARY SYSTEMS

G. W. Preston

January 1981

Prepared for
Office of the Under Secretary of Defense for Research and Engineering

INSTITUTE FOR DEFENSE ANALYSES
SCIENCE AND TECHNOLOGY DIVISION

IDA Log No. HQ 80-2294T

The work reported in this document was conducted under contract MDA 903 79 C 0202 for the Department of Defense. The publication of this IDA Paper does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A098176	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Very Large Scale Integrated Circuits for Military Systems	5. TYPE OF REPORT & PERIOD COVERED Final
	6. PERFORMING ORG. REPORT NUMBER IDA PAPER P-1538
7. AUTHOR(s) G.W. Preston	8. CONTRACT OR GRANT NUMBER(s) MDA 903 79 C 0202
9. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Defense Analyses 400 Army-Navy Drive Arlington, VA 22202	10. PROGRAM ELEMENT PROJECT TASK AREA & WORK UNIT NUMBERS Task T-174
11. CONTROLLING OFFICE NAME AND ADDRESS DUSD Research and Advanced Technology The Pentagon, Washington, D.C. 20301	12. REPORT DATE January 1981
	13. NUMBER OF PAGES 77
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE —

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

None

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Very-High-Speed Integrated Circuits (VHSIC), Very-Large-Scale-Integration VLSI), Signal Processing, hardware macro, switching circuits, future military requirements, FFT, ELINT, Image processing IIR, FIR, serial memory

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The integrated circuit (IC) technology which is to be developed under the ongoing Very-High-Speed Integrated Circuits (VHSIC) program will provide the IC physical characteristics which are necessary for important performance improvements and cost avoidance in many military systems. However, the full realization of the potential benefits of this technology demands new computer architectural concepts based upon new IC functional designs. For the more demanding applications the abandonment of the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. continued

classical "Von Neumann" computer organization (in which instructions and data are stored in the memory, fetched, and processed sequentially) is indicated in favor of processor organizations which permit high degrees of concurrency (parallelism and pipelining), local data storage, reconfigurable data paths (in order to minimize the number of memory fetches), etc. Data processors of this form will require new switching circuits and hardware macros (which perform specialized complex operations). This report reviews some of the available design options for circuits of this type.

IDA PAPER P-1538

VERY LARGE SCALE INTEGRATED CIRCUITS FOR MILITARY SYSTEMS

G. W. Preston

January 1981

INSTITUTE FOR DEFENSE ANALYSES
SCIENCE AND TECHNOLOGY DIVISION
400 Army-Navy Drive, Arlington, Virginia 22202

Contract MDA 903 79 C 0202
Task T-174

ABSTRACT

The integrated circuit (IC) technology which is to be developed under the ongoing Very-High-Speed Integrated Circuits (VHSIC) program will provide the IC physical characteristics which are necessary for important performance improvements and cost avoidance in many military systems. However, the full realization of the potential benefits of this technology demands new computer architectural concepts based upon new IC functional designs. For the more demanding applications the abandonment of the classical "Von Neumann" computer organization (in which instructions and data are stored in the memory, fetched, and processed sequentially) is indicated in favor of processor organizations which permit high degrees of concurrency (parallelism and pipelining), local data storage, reconfigurable data paths (in order to minimize the number of memory fetches), etc. Data processors of this form will require new switching circuits and hardware macros (which perform specialized complex operations). This report reviews some of the available design options for circuits of this type.

ACKNOWLEDGMENTS

The insights, concepts, preferences (and prejudices) expressed in this report, as well as much of the technical material, were gleaned from numerous contacts with the various government agencies and VHSIC contractors. An effort has been made to reference all derived material, but in some cases the actual source has faded from memory. This will serve to acknowledge my debt to the entire community.

Glenn W. Preston

LIST OF ABBREVIATIONS

A/D	Analog-to-digital
AGC	Automatic Gain Control
A/J	Anti-jam
ASP	Advanced Signal Processor
AU	Arithmetic Units
CAD	Computer-Aided Design
CMOS	Complimentary Metal Oxide Semiconductors
CORDIC	Coordinate Rotation Digital Computer
ECM	Electronic Countermeasures
ELINT	Electronic Intelligence
ERIM	Environmental Research Institute of Michigan
ESM	Electronic Surveillance Measures
EW/ESM	Electronic Warfare/Electronic Surveillance Measures
FF	Flip Flop
FFT	Fast Fourier Transform
FIFO	First-In, First-Out
FILO	First-In, Last-Out
FIR	Finite Impulse Response
FTR	Functional Throughput Rate
HOL	Higher Order Language

IC	Integrated Circuits
IIR	Infinite Impulse Response
I/O	Input/Output
IOP	Input-Output Processor
IR	Infrared
JTIDS	Joint Tactical Information Distribution System
LPI	Low Probability of Intercept
LSI	Large-Scale Integration
MIPS	Million Instructions per Second
MPY	Multiply
MSI	Medium-Scale Integration
MTI	Moving Target Indication
POL	Problem-Oriented Language
PSP	Programmable Signal Processor
RAM	Random Access Memory
ROM	Read-Only Memory
SC	Storage Controller
SOS	Silicon on Sapphire
SP	Signal Processor
SR	Shift Register
VHSIC	Very-High-Speed Integrated Circuits
VLSI	Very-Large-Scale Integrated (circuits)

CONTENTS

EXECUTIVE SUMMARY

In 1978, the DoD initiated a major long-term integrated circuit technology program, VHSIC (very-high-speed integrated circuits), under the direction of OUSDR&E for whom this study was performed.

The integrated circuit technology which is to be developed under the VHSIC program will provide the physical characteristics for circuits having two to three orders of magnitude greater data processing capability than then current circuits, but with comparable power consumption, size, weight, failure rate, and manufacturing cost. This is made possible by the use of multiple layers of interconnections, reductions in the dimensions of all elements on the silicon substrate, and very-large-scale integration (VLSI)--i.e., tens of thousands of logic gates. VHSIC circuits would provide the means for significant cost avoidance and performance upgrades in many current military systems, and would remove some of the technical and economic barriers to the introduction of important new systems capabilities (autonomous missiles, for example). But, on the obverse side, at the VLSI levels, difficulties are encountered in circuit layout; design verification; transient conformity; testing; radiation hardness; failure analysis; and user support, including software and systems development aids. Furthermore, VHSIC devices must have much higher logic gate-to-pin ratios than current circuits, and therefore embody functions with numerous levels of logic. Altogether, the pluses and minuses of VHSIC technology have motivated a far-reaching reexamination of computer architecture (Ref. 4).

One of the crucial technical issues for the VHSIC program is the specific circuit functions which are to be developed (product definition). This report reviews some of the technical issues involved in the selection of a set of VHSIC circuits for military systems and recommends certain classes of functions. In some cases, fairly specific circuit functions are described, but these are included for illustrative purposes, not as specific recommendations.

The chip set to be developed must be based on the anticipated usage in military systems, particularly on the commonality of circuit utilization in the various systems. This in turn is best explored in terms of the underlying algorithms to be executed (see p. A-6 for a list of algorithms and the classes of military applications in which they occur). The embodiment of these (and other) algorithms in integrated circuits is treated in this report (page 16 et seq.).

It is a general premise of this study that signal-processing systems based on the classical (Von Neumann) computer architecture do not effectively exploit VLSI technology and cannot meet the more demanding system performance goals of the VHSIC program. Instead, it is recommended that the VHSIC program include the development of hardware macros*, special switching and memory circuits, and other circuitry which is applicable to computer architectures having high degrees of concurrency (pipelining and parallelism) and which minimize the number of memory access cycles.

Three classes of hardware macros are discussed: (1) the dedicated single-purpose macro (such as the complex multiplier or fast Fourier transform butterfly**), (2) dynamically

---

*The term hardware macro refers to circuits which perform a special function (such as correlation, evaluation of a mathematical function, sorting of a body of data, etc.).

**The FFT is executed by a series of butterfly operations and data transfers. The butterfly consists of a complex multiplication and two algebraic additions.

programmable macros (several possibilities are suggested), and (3) firmware controlled ("parameter" or "semi-programmable") macros (such as fixed coefficient filters). Certain special memory circuits (specifically a serial memory in which the decoder is replaced by a shift register, and a decoder which addresses various points on a shift register string) are also described which shorten the memory access cycle for serial data and replace data fetches with shift register rotations.

Programmable switching circuits (cross bar switches, "packet" switches, multiplexers, and demultiplexers) and programmable logic circuits are also cited as potentially valuable uses of VHSIC technology.

This group of circuits, taken together, are seen as offering relatively high payoff in relation to risk and are applicable to various architectures ranging from the hardwired, dedicated configuration to the programmable processor-centered system. The programmable micro processors and micro computers as the building blocks of a distributed network offer similar advantages; they are more flexible than the hardware macro at the cost of lower functional throughput rates.

Finally, unless the VLSI chip set is complete, the designer will be forced to use large numbers of small- and medium-scale integration "glue" chips (to interface and interconnect the VLSI circuits) and the system advantages of the VLSI will mostly be lost*. In other words, to realize the promised system benefits, VLSI must be used throughout (Ref. 33). To some extent, this will require the use of customized circuits, for which the gate array, standard cell, and matrix logic techniques (storage logic array, programmable logic array, associative logic, etc.) are suitable.

*This is all too often the case now at large-scale levels of circuit integration.

This study is a continuation of previous efforts which dealt with the economic motives for the development of VLSI circuits for military systems (Ref. 1) and with their specific applications, which established the functional throughput rate requirements for various systems (Ref. 2).

I. INTRODUCTION

At the chip level the objectives of VHSIC have been stated in terms of Functional Throughput Rate (FTR), which is the product of the number of equivalent gates per circuit and one fourth their switching rate--an eventual goal of 10^7 MHz gates as compared to 2×10^4 MHz gates--the latter corresponds roughly to the best current commercial practice. This increase of about three orders of magnitude is made possible by the use of multiple layers of interconnections, reductions in the dimensions of all elements on the substrate (hence the interest in lithography and related aspects of manufacturing technology), and very-large-scale levels of circuit integration.

In the culminating phase of the VHSIC program, a set of very-large-scale integrated (VLSI) chips will be developed for use by military systems designers in general (Ref. 5), and in demonstration systems (which have been selected by the services--see Table A-1, Appendix) in particular. These will comprise tangible and visible achievements of the VHSIC program. The number of different VLSI chips which can be developed and supported is limited by schedule, resource, and budgetary constraints to a few tens of circuits. This places a premium on the efficacious selection of their functional and physical characteristics. The selection of a chip set (product definition) at the VLSI level, must take account of a wide range of diverse considerations and is generally regarded as one of the most difficult phases in the VHSIC program.

At the higher levels of circuit integration (several thousand gates per circuit) most existing circuits (excluding

memories) are either customized to a single application, general-purpose programmable (such as the microcomputers), or are dedicated to a specific function (the hardware macro, such as the multiplier). Programmable macros which can perform one of several functions have also been proposed and are further explored in this report.

The fully customized circuit designed to a specific application at the highest levels of circuit integration becomes a "system on a chip." With the current state of computer-based design systems, this approach is not available to military systems designers because of inflexibility, the high design cost, long development schedules, risk of design failures, cost of documentation and special test equipment, and risk of subsequent logistics failure; although for equipment such as missiles, which are stored until used, this customized approach seems appropriate (if the shelf life of the IC assemblies equals that of the other components). Computerized design aids may some day be brought to such a level of proficiency that the cost, schedule, and risk are all substantially reduced and systems changes could be economically accomplished at the chip level. In any case, this approach (customizing the chip to the system) requires the use of simplified integrated circuit design rules, which incurs some performance penalty (compared to "leading-edge" handcrafted circuits). These may not always be acceptable--particularly for the high-performance requirements which motivate the VHSIC program (Ref. 6).

This report, therefore, focuses on the chip set which may be developed eventually under the VHSIC program--in particular programmable signal processors (PSP), hardware macros, switching circuits, and specialized data storage and retrieval circuits.

The appealing features of programmable systems at the performance level are versatility and adaptability, and at the economic level the low cost that accrues from volume production.

This latter may seem inconsequential, since compared to other components of military systems, the cost of integrated circuit assemblies is relatively modest in most cases, but the use of circuits which are produced in quantity brings substantial benefits in industry support (system development, support equipment, second sourcing, technology upgrades) and in the reduced cost and schedule for documentation, qualification, training, logistics, etc. The obverse side of programmability is the initial cost of developing software support and system development aids and the continuing cost of developing applications software. Furthermore, in the device itself, programmability is achieved at the cost of extra levels of control circuitry and interconnects which constitute an overhead burden on silicon substrate area and, depending on the application, often result in underutilization of computational assets. Nevertheless, where signal processing elements are embodied in complex systems, programmability is necessary for (1) making upgrades to improve processing techniques or accuracy, or to comply with other hardware changes, or (2) new developments to counter new threats, interface with a new system or sensor, or comply with an overall platform upgrade (Ref. 7).

The hardware macro, on the other hand, represents an extension of the standard parts concept into the LSI and VLSI realm. In a leading edge embodiment, complex but specialized operations can be performed at the highest speed of which the circuit technology is capable. The hardware macro can be easily integrated into micro computer-controlled systems and generally does not require excessive I/O ports (Refs. 8, 9, 2).

In addition to the dedicated, single-purpose hardware macro, several different forms of programmable macros have been proposed: the dynamically programmable macro which executes one of several macrofunctions in response to a control signal, or the firmware-controlled macro in which the macrofunction to be executed depends on the contents of on-board read only

memory (ROM). The latter are also referred to as parameter programmable or semi-programmable. This type of macro is particularly well suited for fixed coefficient arithmetic.

The potential contribution of the hardware macro to a system (as an alternative to executing the same operation from software) can be calculated in terms of functional throughput per chip. This analysis reveals an unexpected synergism among a collection of hardware macros--a bonus in functional throughput (Ref. 9). The hardware macro is also the basis for functional partitioning, in which the processing system is partitioned into a group of standard macros with customized interconnections. The hardware macro is undoubtedly a fruitful source of effective VLSI design, and is the focus of many signal processing studies aimed at identifying the most useful functions. Much of the following report is devoted to this subject.

II. SIGNAL PROCESSORS

The term signal processor (SP) encompasses a collection of computational systems whose chief distinguishing characteristic is their operation on data streams from external sources which must be processed and reacted to in real time. Nowhere are these types of processes more important than in military systems. Missiles and submunitions have become increasingly sophisticated in their use of guidance and sensors, while defense focuses more and more on frustrating the sensor, guidance and control apparatus of offensive weapons through the use of active countermeasures, and electronic countermeasures (ECM). The available reaction time for guidance and control on the one side, or electronic countermeasures on the other may be measured in milliseconds. The signal processor figures prominently in all of these functions: analysis of sensor data, guidance and control, ECM, anti-jam (A/J) stratagems, and so on.

Military signal processing refers in general to the analysis of data originating from sensors (optical, IR, acoustic, radar) from the enemy's emanations (ELINT-ESM), from communications sources [the human voice, Joint Tactical Information Distribution System (JTIDS)]. The processing itself is often arithmetically intensive (and complex in other ways) and is paced by the bandwidth of the signal and external events of the engagement. It is well adapted to pipelining and other forms of concurrency. There is neither time for--nor purpose in--data-dependent branching which figures prominently in data processing. The various types of signals [acoustic, radar, communication, sensor, voice, electronic intelligence (ELINT)]

differ enormously in bandwidth and structure. These are some of the factors which distinguish signal processing from other uses of data processing, and which have already evoked new and distinct architectural concepts.

However, as yet, no programmable signal processor (PSP) has appeared which can effectively deal with such a wide range of applications. Indeed, it must now be recognized that the versatile PSP presents intrinsic difficulties in architecture and software support which can be expected to yield only to sustained, innovative effort. In the following pages we will review some of these difficulties and the steps which may overcome them; the motivation being that a PSP at the VLSI level would be widely applicable to the demonstration systems proposed for the VHSIC program.

Military signal processing applications themselves impose specialized and rather distinctive features on the programmable processor. Included among these are:

(1) High-speed processing paced by data flow and the reaction to events in the environment of the system;
(2) Arithmetically intensive processing and other highly specialized algorithms; encoding and decoding, etc., which require blocks of microcode for efficient operation;
(3) Signal conditioning which precedes the signal processing per se and dictates the initial word size;
(4) Transfer of large quantities of data between bulk memory and the processor, sometimes with specialized and complex shuffling operations;
(5) Data operations which may be implemented in standard hardware macros with a great simplification in code;
(6) Control operations (data transfers, macro instructions, sequencing) which require separate (and generally difficult) software;

(7) Great variability in word size (1 to 16 bits fixed point and 16 to 32 bits floating point) and data parallelism;

(8) Relative freedom from data-dependent branching, multiple levels of interrupt, and so on, permitting the use of pipelining and concurrency (of arithmetic processing with data transfer, etc.) without loss of efficiency.

It is hardly surprising that no universal programmable SP for military applications has yet appeared. A programmable SP sized to the largest requirement and containing the resource for all requirements would be underutilized in most applications if not prohibitively large. For this reason, designers turn to modular processors and other forms of distributed and array processing or functionally dedicated processors.

The shortcomings of present PSPs are sufficiently onerous and fall so far short of universal applicability that the most fundamental architectural features of future PSPs are still "up for grabs."

A. TYPES OF SIGNAL PROCESSING ARCHITECTURE

SPs have been designed (and proposed) based on various, radically different organizations of computational resources, the only essential common feature being concurrency, either in the form of pipelining, array processing, distributed processing, parallelism, or combinations of these (Ref. 10). All of these configurations of computer resources may consist of programmable or nonprogrammable processing elements and the interconnections between them may be hardwired or reconfigurable--on command from a central control unit. In any case, the degree to which the processing task lends itself to concurrency can often be clarified by a block diagram (e.g., FFT followed by the computation of magnitude followed by peak detection and thresholding,

etc.). Several examples are given in Figs. 1A and 1E. The potential for pipelining as the signals pass from block to block is readily apparent and the operations in many of the blocks lend themselves to parallel processing (several butterfly processors performing an FFT, for example). On a lower level, the algorithms themselves can often be represented by block diagrams which can in turn be related to processing elements. Examples of algorithm block diagrams are shown below (Fig. 2) for the FFT, adaptive processor, and an infinite impulse response (IIR) filter section.

Each of these algorithms can be computationally realized in many different ways, ranging from a "conventional" Von Neumann computer* (in which the signals at each point in the block would be stored in separate memory cells, then fetched and transferred to a central processing unit and so on, one step at a time) to a "hardwired" processor, a more or less literal embodiment of the block diagram. Signal processing systems usually fall somewhere in between. For example, the SPS-41 contains multipliers and adders which can be configured into more or less arbitrary pipelined arrangements, so that the resulting signal flow corresponds to the "block diagram" of this algorithm.

In the following sections, some of the properties of SPs based on the various configurations will be discussed. In the end, we will focus on two broad types: first, programmable general-purpose processors and second, configurations of dedicated processing elements (functional configuration). The hardware macro (which may itself be reconfigurable under program control or by firmware) will figure prominently in both. Both of these principal types of programmable general-purpose

*...where data and program share the same memory space.

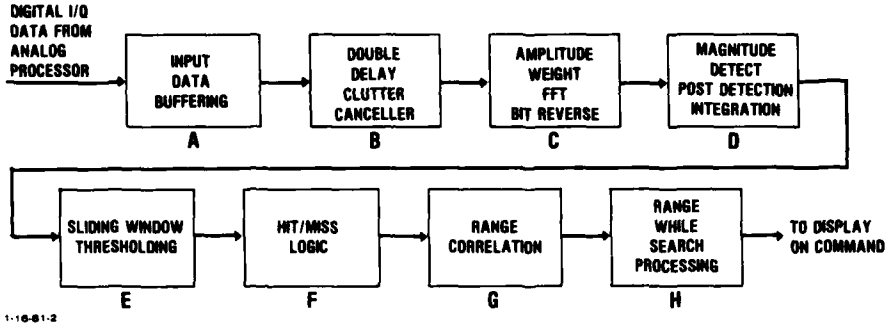


FIGURE 1A. Air-to-air radar signal processor

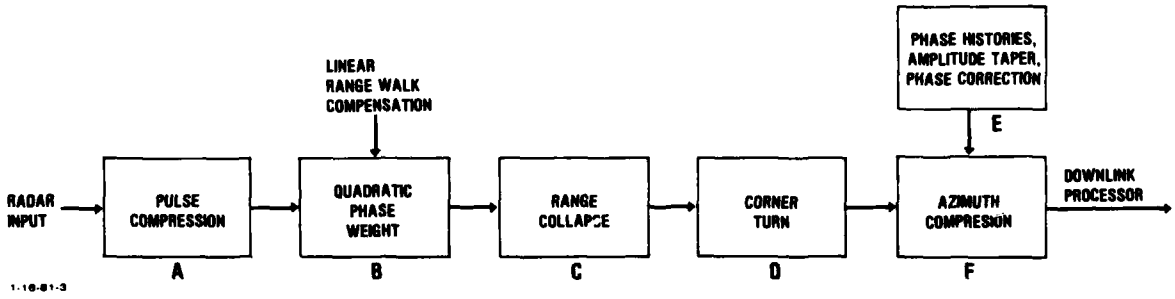


FIGURE 1B. SAR system translation invariant processing

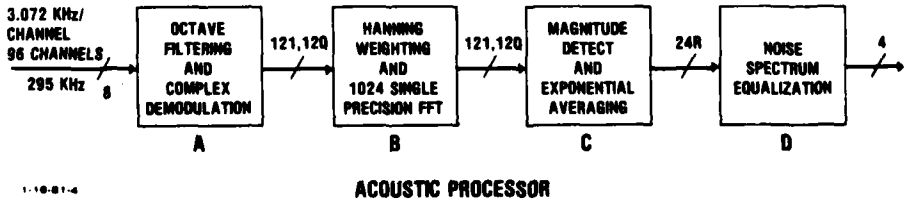


FIGURE 1C. Signal processing block diagrams

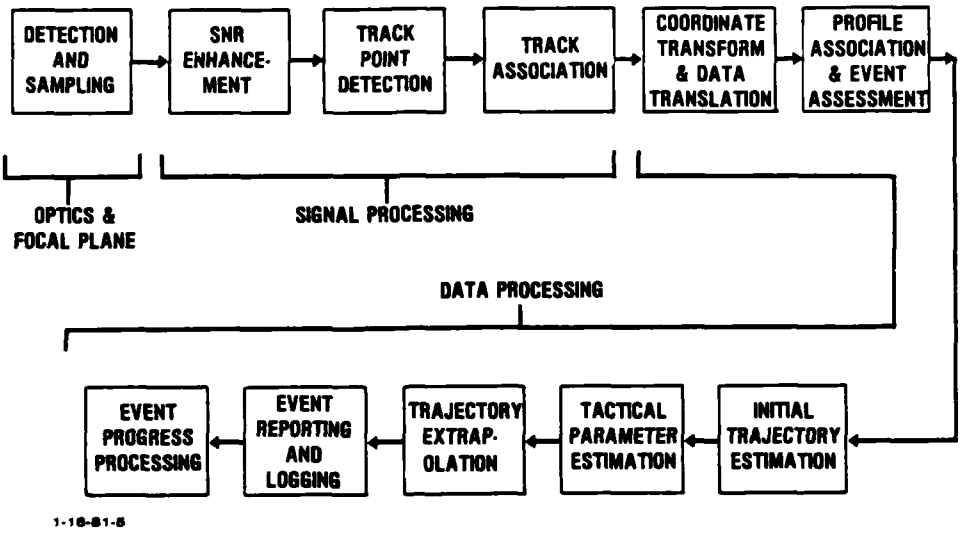


FIGURE 1D. IR surveillance system

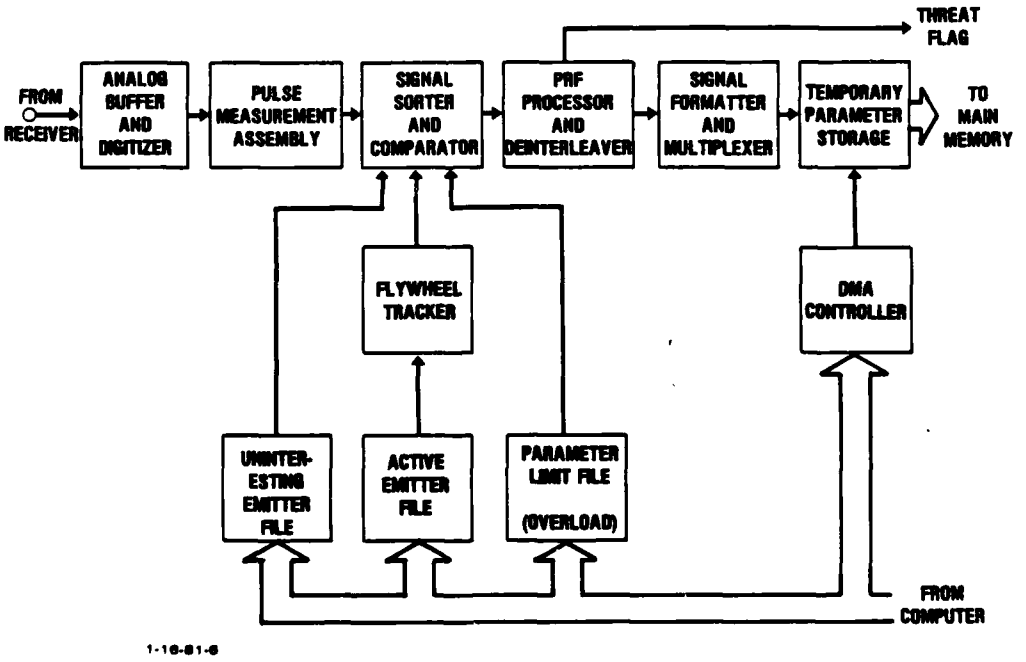


FIGURE 1E. A EW/ESM generic preprocessor

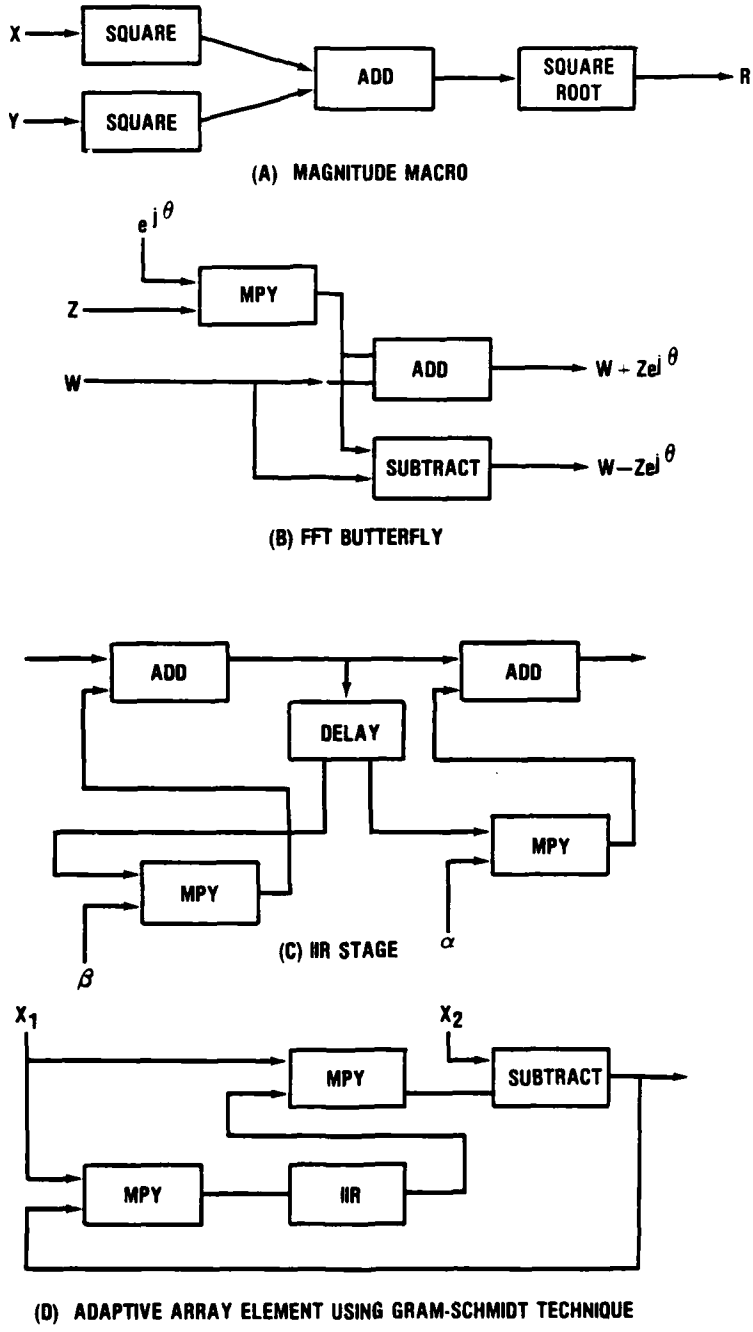


FIGURE 2. Algorithm block diagram

processors and also functionally partitioned configurations, operating in tandem, have been advocated for some SP applications (see Fig. 3).

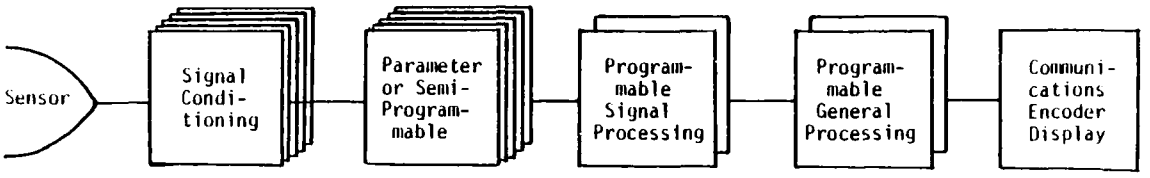


FIGURE 3. System architecture (courtesy IBM)

The hardwired (dedicated) assembly of macros has the very considerable merit of eliminating the control circuitry for complicated data transfers between the processing elements and working storage. The data transfers occur in the customized wiring, and much of the intermediate data storage is eliminated (by providing processing elements for all data streams) and so is the need for generating blocks of microcode (an expensive, error prone, often inefficient and, by most accounts, onerous procedure). This approach requires lavish use of processing resources and is often impractical with current circuitry.

To recapture some degree of flexibility for this approach, the use of programmable structures for interconnecting macros (or for that matter programmable processing elements) are being explored; these include register programmable cross bar switching structures (Ref. 11), "packet switching" structures in which switching instructions accompany the data (Ref. 12), and field programmable switching networks.

Those signal processing operations which actually account for the bulk of the functional throughput capacity are operations such as spectral analysis, adaptive processing, electronic surveillance measures (ESM) signal sorting and analysis (blocks

B and C of Fig. 1A, for example). These consist of highly repetitive arithmetic operations (with little, if any, conditional branching, etc.) for which the programmable general-purpose architecture is ill suited. It is for these applications that the functionally configured approach is most suitable.

Historically, parallel processing has been advocated for arithmetically intensive problems dealing with data having a natural spatial significance, such as image sensor data, boundary value problems, etc. Superficially, parallel processing can be seen as a means of increasing computational rates (given the limitation of circuitry) but for the programmable parallel processor--which must be judged by its efficiency with a mix of applications--no clear-cut advantage has as yet been shown, primarily because the computer resources cannot be fully utilized in the execution of all programs. In any case, processing speed may, for many programs, be less significant than the efficiency and simplicity of data transfers. Balance is rarely achieved with "classical" architectures when applied to signal processing. They often become speed limited by memory transfers, even with two-port memory structures, and the data declarations, indexing, etc., are common sources of software error unless the compiler can make the transfers between working storage and memory efficient and transparent to the user.

Distributed processing usually refers to the use of a network of microprocessors (such as the Texas Instruments Micro Vector Processor) or microcomputers each dedicated to a subtask in the system. This approach exploits the modularity of many signal-processing applications. Also, in this way, programmable processing elements become an alternative form of functional partitioning. The parts count is reduced to the extent that the same type of processing unit performs many functions. On the other hand, the processing elements may not be fully utilized, i.e., functional throughput capacity is sacrificed. The corresponding modularity in software alleviates the central control

problem, but portability is nonexistent. Both the parallel (array) and distributed processing may use "hardwired" data transfers or programmable switching structures between elements.

Parallel processing is also advocated as a possible means of taking advantage of the economics of large-scale integration by utilizing repetitive system organization with other advantages related to fault tolerances, for example (Ref. 13, Chapter 3). However, the various parallel processing approaches have enjoyed only limited success because of (1) programming difficulties, (2) inefficient utilization of computer resources for many applications. As more refined lithographic methods bring faster circuitry, the necessary extent of parallel processing would apparently diminish, except for truly parallel processing applications for which these difficulties do not exist.

Although parallel processing is more written about than practiced, functional parallelism--in which the separate computer functions are performed in parallel--is used for signal processing almost without exception.

The limitations of the programmable signal processor for military applications are painfully apparent; chiefly their inability to deliver the required throughput even with hand-microcoding (for the complex data transfers and concurrent processing) although vigorous efforts to design high throughput programmable processors are continuing (focusing principally on array processing).

Meanwhile, interesting progress is occurring in hardware macros, special memory circuits, and switching circuits, which may profoundly affect future signal processor architectures. This is the result of a number of factors: the plummeting cost of circuitry at higher levels of circuit integration, advances in algorithmic analysis, and continuing innovations in circuit configurations. The cost per function in relation to level of

integration (minimum feature size) is well documented and the past trends are expected to prevail to at least the 1μ region. The progress in algorithmic analysis is exemplified by numerous applications of bit partitioning (Refs. 14, 15), and "merged arithmetic" (Ref. 16), while the list of innovative circuit configurations includes the butterfly switch (Ref. 17), the DIMOND switching circuit (Ref. 3), multiport memories, etc.

B. MACROS

The term macro is used rather loosely to characterize a block of instructions or subroutine in software, or, in this context, a circuit or assembly of circuits--a hardware macro--dedicated to the execution of such a subroutine. In the latter sense, it has become a subject of intense interest to IC designers as a practical approach to the horrendous design problems at the VLSI level and beyond (10K gates and more). It seems apparent that without the use of standardized blocks of circuitry supported with powerful computerized design aids with well-understood physical and logical characteristics the problems of design verification, fault isolation, timing, etc., could well prove to be prohibitive at the VHSIC level of integration. The progress of VHSIC technology seems closely linked to that in computer-aided design (CAD) tools; and a library of macros plays a central role in CAD for VLSI.

The progress to date in defining a group of standard LSI macros has been modest, at best. Most highly computerized design methods are based on rather small blocks of circuitry (100 transistors or so) as in the standard cell (Ref. 18) and the CALTECH "silicon compiler" methods. Attempting to extend the standard cell concepts to larger blocks of circuitry would run afoul of the Smith-Fubini relation (Ref. 19), which forecasts the need for a very large number of different cells with little commonality of application. Nevertheless, new hardware

macro designs at the VLSI level are continually being introduced (the multiplier, multiplier and adder, correlator, FFT butterfly, divider, the "MATH" chips) and the matter is being widely and vigorously pursued under the VHSIC program. The contribution of this approach to future VLSI technology remains to be seen. The negative attitude sometimes expressed toward the use of standard hardware macros (Ref. 20) may reflect a misconception of the military applications; since the macro approach is very well suited to military signal processing in which (as we have already seen) a relatively few algorithms and macro functions comprise the bulk of the functional throughput rate.

In fact, referring again to the algorithmic block diagrams of Fig. 2, we see that only three macros would be needed for a hardwired embodiment of all four; namely a multiplier (particularly a complex multiplier), an adder/subtractor, and a square root evaluator.

The term macro is also often applied hierarchically to entire algorithms such as the FFT, linear predictive coding, and so on, although these could not be implemented in monolithic circuits except with fractional micrometer lithography. Instead, these macros would be executed by a block of instructions using a programmable processor, possibly in conjunction with hardware macros or by a hardwired assembly of circuits. The high level macros are the key to identifying the standard macros from which to assemble future systems.

A considerable number of macros have been identified (Refs. 8, 2) at various levels in the hierarchy.

At the highest level, we have (among others):

- Filtering
- FFT
- Adaptive processing (antenna arrays, MTI, CFAR)
- Linear predictive coding

- Image processing algorithms
- Error correction encoding and decoding
- Track association and smoothing (Kalman filtering)
- Ambiguity resolution.

The functional throughput rate for each of these varies considerably, depending on the parameters for the application (bandwidth, spectral resolution, degrees of freedom, etc.).

At a lower level where a monolithic embodiment may become feasible, the identified operations include:

- Magnitude
- Phase
- Complex multiply
- Peak, median, etc., of data
- Correlation
- Division
- Associative memory (content addressable)
- Sort and merge memory
- Multiport memory
- FIFO stack
- Barrel shifter
- Floating point conversion
- FFT butterfly
- Sine/cosine generator
- Logarithm, exponent generator
- Histogram
- Matrix transpose
- Data reordering
- Programmable frequency synthesizer
- Interconnect matrix.

In the following section, several macro functions, switching circuits, and memory circuits are examined for the purpose of illustrating the various design principles and architectural features. The hardware macros include examples of bit partitioning,

merged arithmetic, and programmability; several interesting and innovative switching circuits are described and two cases of special memory circuits are given. In certain cases, logic designs are proposed which may have some practical utility for VHSIC. However, all of these design concepts are offered on the basis of their logical structure. No detailed circuit embodiment was considered, so that the practical merits (if any) of these circuit concepts with respect to speed, power consumption, or circuit density remain to be evaluated.

C. COMPLEX MULTIPLY (MPY)

The complex multiply operation (the cornerstone of signal processing) would find application in the FFT, adaptive processing, band shifting, and filtering, in addition to purely arithmetic operations.

Since

$$(u + iv)(x + jy) = (ux - vy) + j(vx + uy)$$

there are four real multiplies and two algebraic additions.* Because of the large number of gates involved in the multiplication of 12- to 16-bit numbers, the four full multipliers and two adders could be placed on a monolithic substrate only if the circuit dimensions were close to 1μ.

Another practical difficulty is that of pin-out. If the multiplications were performed with full precision and separate I/O ports were provided for all four real inputs and the two real outputs, a total of 8B pins would be needed for the data, or 118 for 16-bit words.

This macro would find very wide application in military signal processing systems.

---

*...or, by Golub's method, five adds and three multiplies.

D. DATA SORTING CIRCUITS

1. The Pipeline FFT

Since spectral analysis figures so prominently in military signal processing, a great deal of effort has gone into modifications of the fast Fourier transform (FFT) algorithms to make the computation utilize silicon resources most efficiently. Also, special architectures have been devised which match memory cycles to processing cycles and thus avoid idle time (Refs. 21, 22). This is accomplished by pipelining some number of processors, depending on the processing time (of a butterfly, for example) relative to a memory cycle. The circuit configuration described in the references deserves consideration for a VLSI embodiment.

Some of these circuits are intended to simplify and accelerate the reordering of data (between successive columns of butterflies) to make efficient use of limited computational resources (for example, a single butterfly processor). Figure 4 schematizes the data flow for the radix 2, N = 16 FFT. A straightforward configuration for using one butterfly processor would consist of two random access 2N word memories. With the N (complex) datum stored in one of the memories, the input data to the butterfly processor are accessed in the indicated order (X_0 and X_8, the X_4 and X_{12}, etc.), and the results y_1, y_2, then y_3, y_4, etc., stored in the second memory. On the next cycle the second random access memory (RAM) is read out to the butterfly processor in the order (y_1, y_3, then y_2, y_4, etc.) and stored in the first RAM, and so on. The sequence in which the data is accessed would need to be computed or stored in a separate memory, then read out and used to address the RAMs containing the y's.

The computation of each butterfly would consist of one read cycle, one processing cycle, and one write cycle, although the "read" and "write" cycles can overlap. The total cycle for

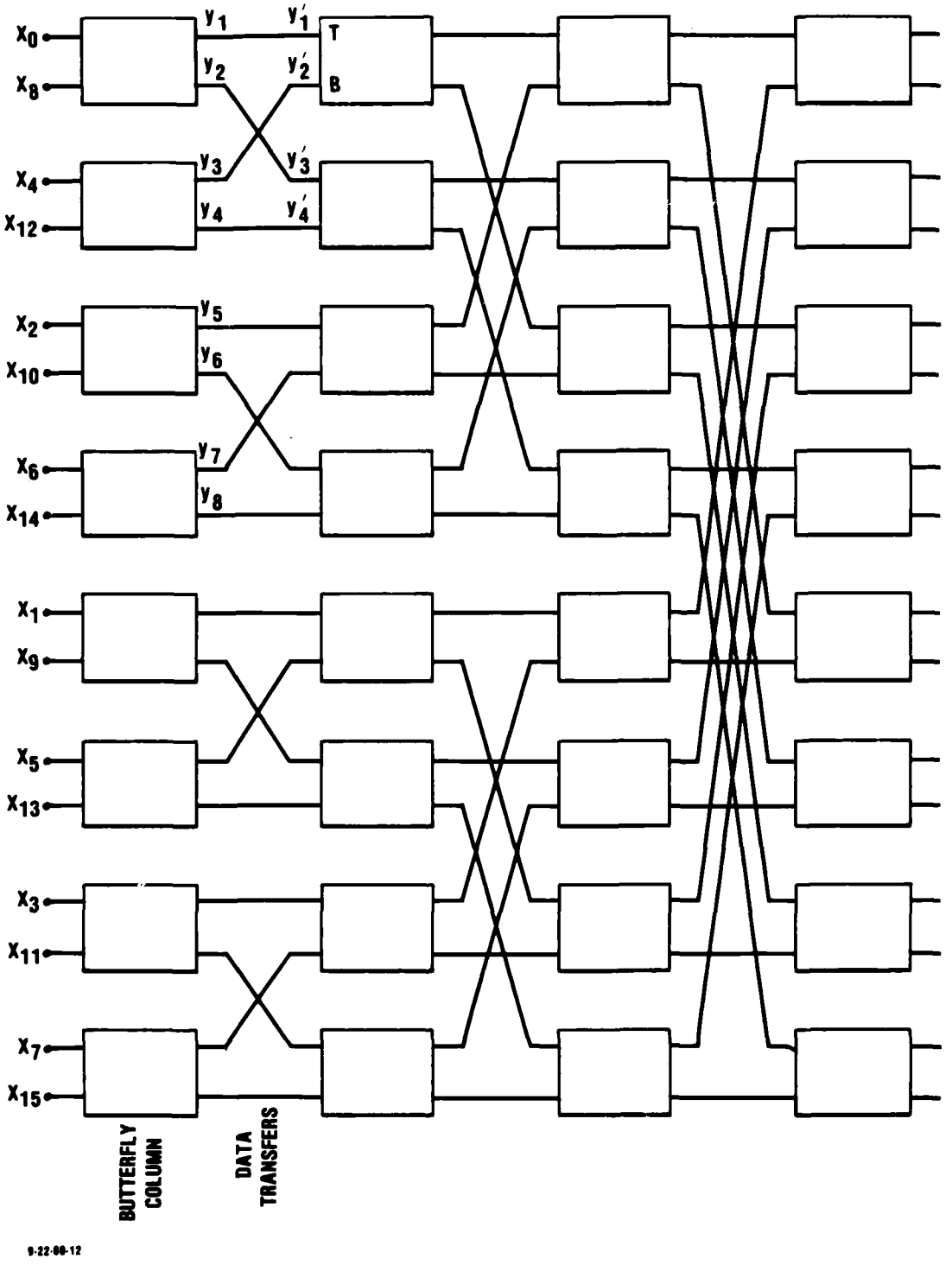


FIGURE 4. Schematic of 16 point, radix 2, FFT

the butterfly would equal the length of the read cycle plus the processing cycle unless the write cycle exceeds their sum.

An alternative configuration for switching and storing the data between butterfly columns which is closely related to those described in Refs. 21 and 22 will now be described. It would appear to result in shorter "read" and "write" cycles, particularly the former, and also simplifies the sequencing of the ROM fetches.

To illustrate the data transfer more clearly, assume that the butterfly processor always works down the column, taking its input in successive pairs. Denote input data by primed and the output data by unprimed symbols, the upper input to the butterfly by T, and the lower output by B.

On the first column the processor would take (x_0, x_9) and output (y_1, y_2), then (x_4, x_{12}) and output y_3, y_4, etc. But when executing the second column $y_1 \rightarrow y_1'$, $y_3 \rightarrow y_2'$, $y_2 \rightarrow y_3'$, $y_4 \rightarrow y_4'$, etc. Similarly, when executing the third column $y_1 \rightarrow y_1'$, $y_5 \rightarrow y_2'$, $y_3 \rightarrow y_3'$, $y_1 \rightarrow y_4'$, $y_2 \rightarrow y_5'$, $y_6 \rightarrow y_6'$, $y_4 \rightarrow y_7'$, $y_8 \rightarrow y_8'$, etc., and for the third column $y_1 \rightarrow y_1'$, $y_9 \rightarrow y_2'$, $y_3 \rightarrow y_3'$, $y_{11} \rightarrow y_7'$, $y_5 \rightarrow y_5'$, $y_{13} \rightarrow y_6'$, $y_7 \rightarrow y_7'$, $y_{15} \rightarrow y_8'$, $y_2 \rightarrow y_8'$, etc.

These transfers are schematized in Fig. 4.

After the column of butterflies has been executed the data are transferred up or down by one position or not at all; after the second column up or down by three positions or not at all, and so on.

It can now be seen that both the read and write cycles could be shortened by a special memory device referred to here as the "P" box (see Fig. 5) consisting of two shift register strings (one for the T inputs, one for the B) fed laterally by

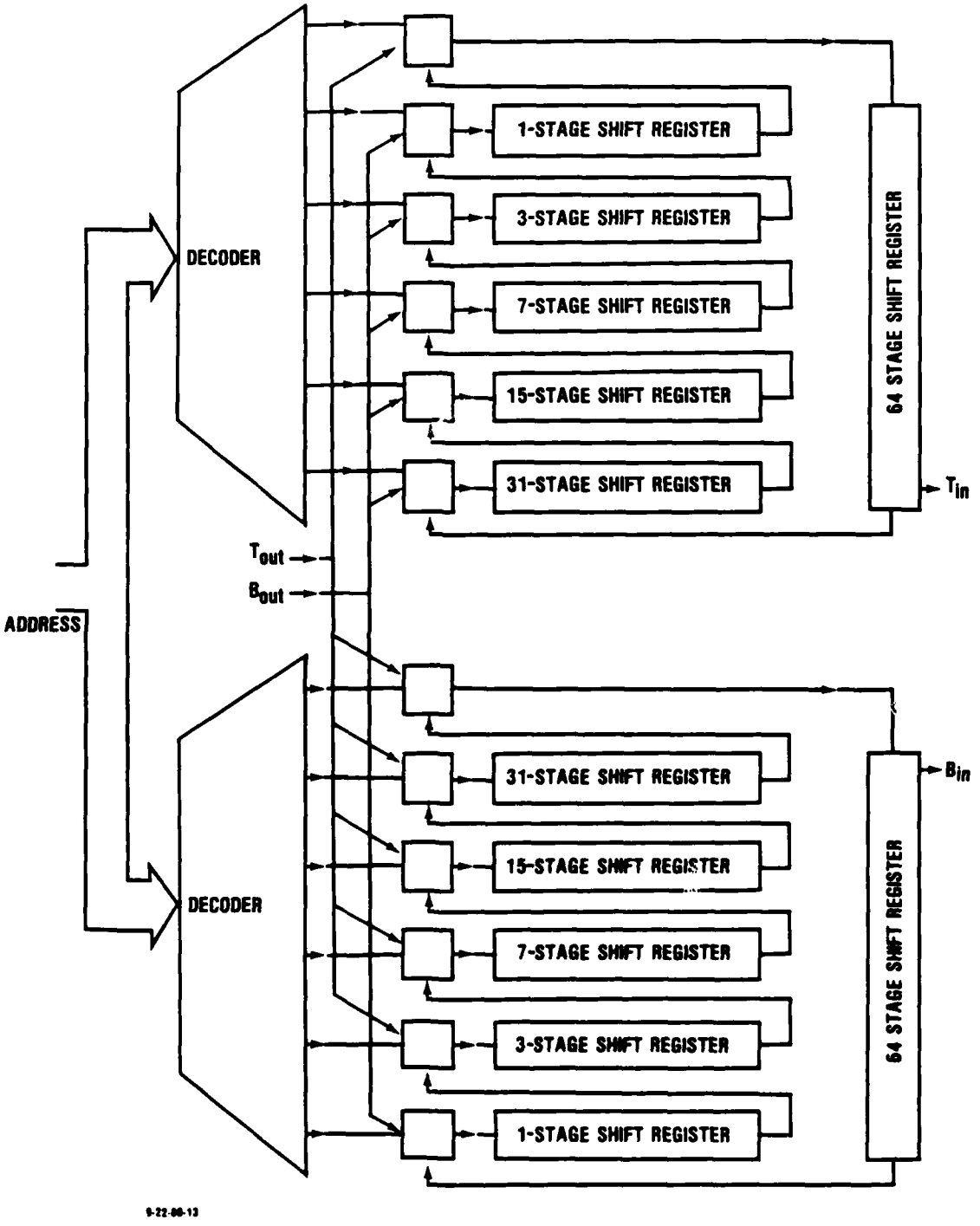


FIGURE 5. "P" box

e decoder which places the bits in the shift register either 0, 1, 3, 7, etc., positions removed from the "current" position. As the butterfly processor reads the shift register string from the previous column it places the output in the shift register string through a decoder which shifts the position of the stored data according to Fig. 4 so that the data is in the natural sequence for the next cycle. The write cycle is shortened because the decoder net which selects the bit position has only $\log_2\{2\log_2 N-1\}$ levels of decoding rather than $\log_2 N$. The read cycle is shorter because the address is generated by a shift of the shift register string of one position.

When the data is repositioned, a T output becomes a B input and vice versa. If greater speed is needed it can be achieved by the use of several butterfly units operating in parallel, the P box being split into a pair of loops for each butterfly. The total number of shift register stages equals N for all such configurations, but the speed would increase in proportion to the number of butterfly units.

The control sequence for the P box is quite simple. For a given column of butterflies, all data which are shifted in position (either T or B outputs) are shifted by the same number of steps (1 or 3 or 7 or 15 ...), in general, 2^J-1 for the J'th column of butterflies. The controls then would consist of $\frac{N}{2}$ bits for each column (a total of $\frac{1}{2}N\log_2 N$) which determines whether a given output is shifted or not. The address is either zero or 2^J-1.

The shift register memory could be replaced with conventional RAM circuits, in which case the address would then be the count of the level in the column at which the butterfly is operating to which the quantity 2^J-1 is added for those datum which require repositioning.

The P box structure might be useful in other applications which involve bit shuffling. Shift register strings of various (and variable) lengths can be built up from standard ROM circuits (Fig. 7).

2. Signal Sorting, Streak Detection, Track Association

Signal sorting, on the basis of pulse repetition frequency, utilizes a considerable amount of signal processing resources in electronic warfare/electronic surveillance measures (EW/ESM) equipments (Ref. 23); in lieu of an adequate automatic processing capability, the function is now performed manually (Ref. 24), which involves a human operator, displays, etc., and a sacrifice in performance (acquisition speed, saturation signal density).

Various automatic processing techniques, such as the histogram and spectral analysis, have been advanced for this purpose.

The detection of lines in an optical image is very similar, in principle, to the identification of a series of radar returns from a moving target.

A special circuit, for these purposes, which might be structured along the lines outlined below, would appear to be less complex (require less silicon resources) and possibly be more sensitive and accurate than either the histogram (which is subject to ambiguities) or spectral analysis (an exorbitant consumer or processing resources).

In essence, the device tests for the coincident occurrence of pulses at various fixed intervals. The circuit, as shown in Fig. 6, tests for the occurrence of pulses at three successive intervals of either N-1, N, or N+1 clock cycles. When all of the data has cycled through the four shift register strings, the lengths of the register are all shortened N→N-3 and the process repeated until the desired range of interpulse periods

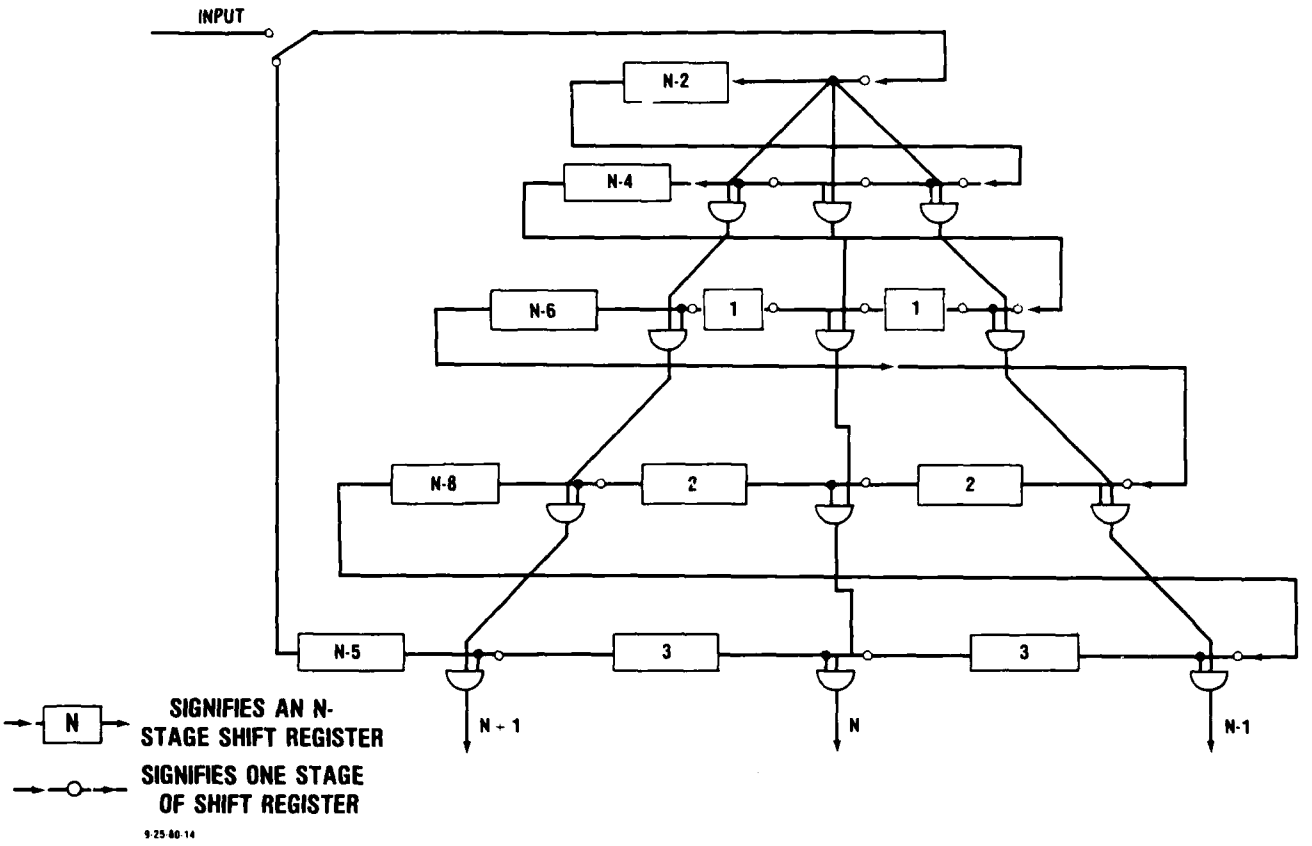


FIGURE 6. Memory circuit for streak detection in image processing, pulse period sorting in EW/ESM

has been searched. The programmable shift register (Fig. 7) could be used for the five shift register strings. As shown, one bit is stored for each cell; the extension to multiple bits words is straightforward. This type of circuit may be extended in various ways to increase accuracy and throughput, such as several delta networks in series testing for successively larger (or smaller) pulse periods (this would proportionately increase throughput rate). Similar circuitry has been constructed at the Environmental Research Institute of Michigan (ERIM) (Ref. 25).

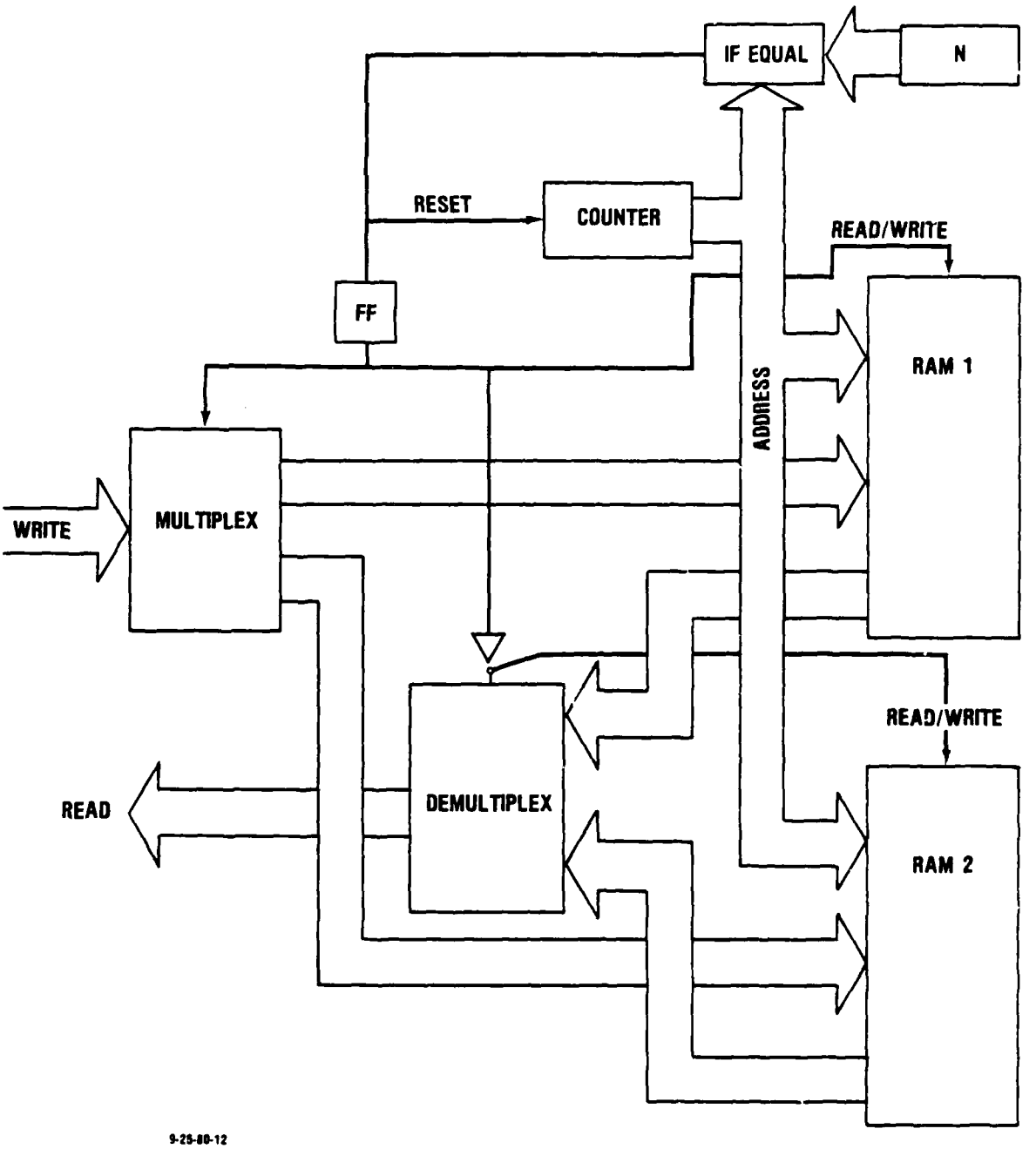


FIGURE 7. Programmable shift register stack

A dynamically variable multi-bit shift register stack is shown schematically in Fig. 7. During a given cycle, data is written into one of the RAMs and read from the other. When the number of data reaches the desired shift register length (N), stored in a separate register, the counter is reset and the role of the RAMs interchanged so that the one which was written onto during the previous cycle is now read from, and vice versa. The read/write operations occur simultaneously and the length of the stack can be changed by entering a new number into the register. Of course, the maximum number of shift register stages cannot exceed the number of words of the RAM sections.

E. FIXED COEFFICIENT ARITHMETIC (THE IIR FILTER, MULTIPLICATION)

A good example of the firmware-controlled macro is fixed coefficient arithmetic; for instance, a multiplier circuit. In principal, any arithmetic operation can be embodied in the form of precomputed stored tables, but these are not generally useful. A 12 x 12 multiplication table would contain $24 \times 2^{24} \approx 400$ Mbits. A ROM system of this size would be incomparably slower and more expensive and power-consuming than a commercially available 12 x 12 monolithic multiplier; but a table of fixed coefficient multiplication would require only $24 \times 2^{12} \approx 100$K, which would be comparable in speed and cost and would consume less power (assuming comparable design rules for the ROM and multiplier). The ROM space needed for fixed coefficient multiplication can be greatly reduced by separating the input into two words consisting of its most significant half and its least significant half and applying them separately to one (or two) ROMs and adding the result (Ref. 15). In this way the product of a 16-bit word by a fixed 16-bit coefficient needs $32 \times 2^8 =$ 8K ROMs, but another layer of logic (the ADDER) has been added. This increases latency somewhat, but does not necessarily reduce the throughput rate.

A more interesting example of fixed coefficient arithmetic (based on the use of binary decomposition) is the method of Peled and Liu for the IIR filter, or any linear function of a group of variables (Ref. 26). Linear operations on sampled data can be factored in a number of ways, such as a cascade of operations of the form:

$$H(z) = \frac{1 + \alpha z^{-1}}{1 + \beta z^{-1}} .$$

The corresponding recursive formula for the output Yn (on the n'th cycle in terms of the input sequence Xn) is $Yn = Xn + \alpha Xn-1 - \beta Yn-1$. The block diagram of this operation (shown in Fig. 8A) translates directly into hardware embodiment (Fig. 8B).

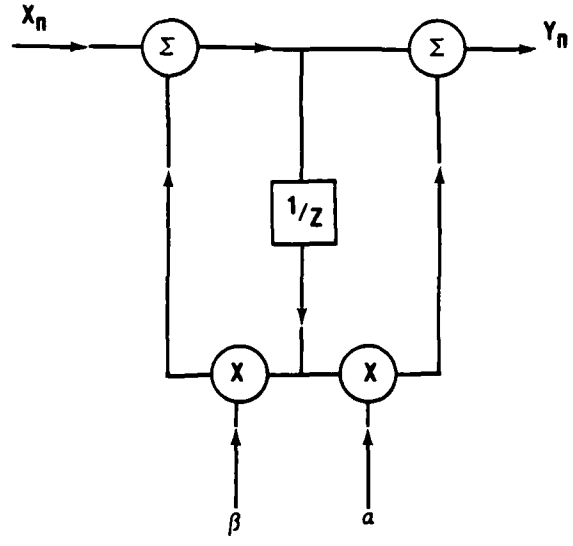
An examination of the input-output relationship, in terms of the bit sequence representing each variable, illustrates a more general approach (binary decomposition):

Suppose $Xn = X_o \sum_{j=0}^{B-1} X_n^j 2^{-j}$ where $X_n^j (= 0, 1)$ is the bit sequence, and $X_o = 2^k$ --a scale factor. Then

$$Y_n = X_o \sum_{j=0}^{B-1} (X_n^j + \alpha X_{n-1}^j) 2^{-j} - Y_o \beta \sum_{j=0}^{B-1} Y_n^j 2^{-j}$$

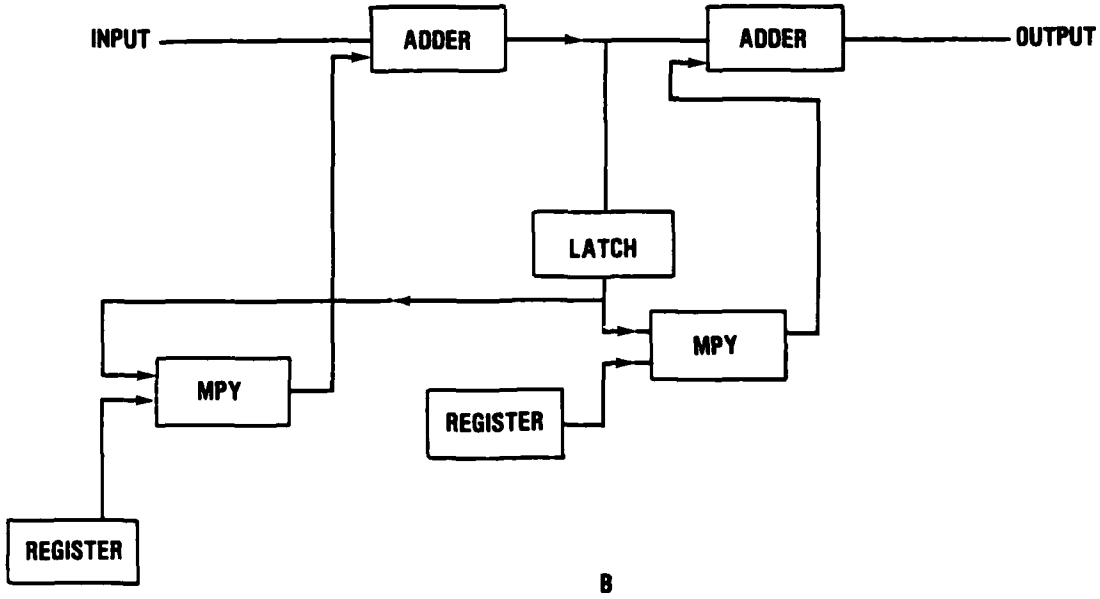
$$= \sum_{j=0}^{B-1} \left[X_o (X_n^j + \alpha X_{n-1}^j) - Y_o \beta Y_n^j \right] 2^{-j}$$

$$= \sum_{j=0}^{B-1} \phi(X_n^j, X_{n-1}^j, Y_n^j) 2^{-j} \equiv \sum_{j=0}^{B-1} \phi(j) 2^{-j} .$$



ALGORITHM BLOCK DIAGRAM

A



B

STRUCTURAL BLOCK DIAGRAM

9-22-80-10

FIGURE 8. Pole-zero filter stage

The function ϕ of 3 bits could be stored in an 8-word memory, accessed B times by the successive bit sequences X_n^j, X_{n-1}^j, Y_n^j, and the results shifted and added according to the above formula. This is a form of "merged arithmetic," i.e., the multiplications and additions are merged (Ref. 16). However, this does not give direct comparability with the hardwired embodiment because B cycles of memory access and addition are required. This might be reduced to one cycle by the use of 7B ROMs (identically coded), and operating in parallel and followed by an adder tree. The adder tree is more complex than the adders of the hardwired embodiment but the two multipliers are disposed of and the amount of memory used is inconsequential. Alternative configurations allow for tradeoff between the complexity of the adder network and the ROM space.

For example, the sum of several ϕ terms can be stored:

$$\phi(X_n^1, X_{n-1}^1, Y_n^1, X_n^2, X_{n-1}^2, Y_{n-1}^2) = \phi(1) + \phi(2) .$$

In general, if the sum of n term is stored in each ROM, a total of $\frac{B}{n}$ ROMs are needed, each of which holds $B2^{3n}$ bits. The number of adders following the ROMs is $\frac{B}{4n}\left(1 + \frac{B}{2n}\right)$.

n	ROM Space	Number of Adders
1	12(8 x 12)	21
2	6(64 x 12)	6
4	3(4K x 12)	2

For n = 4 the ROM embodiment becomes directly comparable with the hardwired direct form and would appear to offer a useful reduction in cost and power for equal speed. Incidentally, the ROM space for n = 4 is actually less than for the fixed coefficient multiplication by ROM look-up.

The speed advantage enjoyed by the ROM look-up method is not fully apparent from a comparison of the ROM memory cycle (and the following adder chain) with the sum of the adder and multiply delays. The cycle time of the direct (hardwired) embodiment would have to exceed the sum of the MPY cycle and twice the ADD cycle before the loop stabilizes. In general, embodiments of the Peled and Liu look-up method appear to simplify timing problems in comparison to a "literal" embodiment.

F. CORDIC

We consider here a computational technique known as CORDIC (the COordinate Rotation DIgital Computer) (Ref. 27, 28). CORDIC was originally conceived for the purpose of computing the magnitude and angular argument of a vector (given the rectangular coordinates) or for computing the coordinates of a vector after rotation. It was subsequently extended to decimal binary conversion (Ref. 14 and the computation of the functions of a single variable (circular and hyperbolic functions, logarithm, exponential, and square root) and functions of two variables (product and ratio).

The CORDIC method is based on a series of rotations by a sequence of angles $\alpha_i = \tan^{-1} 2^{-(i-2)}$. By this means the process is reduced to the operations of shifting and algebraic addition. On the first step, the vector (having initially the components X, Y) is rotated towards the X axis (say) by 90°. On the next step, the resultant vector is rotated toward the X axis by 45°; then the resultant vector is rotated toward the X axis by 22.5° and so on. The polarities of the successive rotations ($\varepsilon_1 = \pm 1$) being determined by the net rotation to that point.

During the process, the magnitude of the initial vector is lengthened by a fixed constant (which is characteristic of the number of such rotations--provided none are omitted--independent of the polarities ε_1). When the sequence of rotations has brought

the resultant vector to the X axis (say) (within a predetermined error) then the initial angular argument is given by the net sum of the rotations and the magnitude by the adjusted final value for X.

In detail, the arithmetic steps are (for the ith step)

$$Y_{i+1} = Y_i + \varepsilon_i 2^{-(i-2)} X_i$$

$$X_{i+1} = X_i - \varepsilon_i 2^{-(i-2)} Y_i \ ,$$

the final rotation being

$$= \sum_i \varepsilon_i \alpha_i \ .$$

The apparent multiplications $2^{-(1-2)}X_i$, etc., consist of shifts of the bit patterns for X_i, Y_i.

In principle, the sequence ε_i and the final value of X (Y is generally forced to zero by the rotations) could be precomputed for each X, Y, and stored in ROM, but the size of the resulting ROM space would be prohibitive in most cases of interest.

The direct method for calculating R and ϕ would be the evaluation of

$$\phi = \arctan \left\{\frac{Y}{X}\right\}$$

and

$$R = \sqrt{X^2 + Y^2} \ ,$$

which could be performed rapidly if special hardware macros were available for division, the arctangent, and square root functions.

The use of precomputed tables for the square root and arctangent are a distinct possibility, provided no more than 14-bit precision is wanted (each would require about 230K bits).

The CORDIC algorithm involves one cycle of shift and addition (for each component) for eve in the word. This process can be accelerated considerabl by a modification such as the following.

The point of departure for the present discussion is the decomposition of the original components (X, Y) into the sum of several vectors of descending magnitude by grouping the B bits of (X, Y). The rotation of the vectors corresponding to the most significant group of bits of X, Y can then be accomplished by look-up in precomputed tables. In terms of the bit pattern for X,

$$X = 2^N \sum_{i=0}^{B-1} 2^{-i} X_i \ ,$$

the sequence of new vectors $x_n(y_o)$ are the successive groups of n bits of X(Y);

$$X = 2^N \sum_{i=0}^{\frac{B}{n}-1} 2^{-in} x_i(n)$$

$$x_o(n) = X_0 + X_1\, 2^{-1} + X_2\, 2^{-2} + \ldots X_{n-1} 2^{1-n}$$

$$x_1(n) - (X_n + X_{n+1}\, 2^{-1} + X_{n+2} 2^{-2} + \ldots)\, 2^{-n} \ ,$$

and so on.

If the first components of X, Y contained no more than 6 bits (say), then the corresponding angle

$$\phi_1 = \arctan\left(\frac{y_o(6)}{x_o(6)}\right)$$

would be stored in a ROM table containing 16K words.

To yield a useful result in all cases, the ROM table must be entered with the most significant non-zero vector components (X, Y), which means that several different ROMs are needed. To make this more specific, suppose X and Y were represented by 14 bits and each of them was then separated into its most significant 7 bits (x_1, y_1) and the second most significant 7 bits (x_2, y_2). Then, if x_1 and y_1 were both non zero they would be used to enter ROM 1 (say) for ϕ, also if x_1 and y_1 were both zero, then x_2, y_2 would be used to enter ROM 1. On the other hand, if x_1 (but not y_1) were zero then x_2, y_1 would be used to enter ROM 2, and if y_1 (but not x_1) were zero, then x_1, y_2 would be used to enter ROM 3.

The rotation of X, Y through ϕ_1 could be computed directly:

$$X' = X\cos\phi_1 + Y\sin\phi_1$$

$$Y' = Y\sin\phi_1 - X\cos\phi_1 ,$$

which is, in fact, a complex multiplication where the multiplicand has a magnitude of unity. The process is then repeated on X', Y', etc. (This computation might share the resources of an FFT butterfly unit if it were available in the system.)

If, instead of the true angle ϕ_1, the stored value were that α_1 nearest in value to ϕ_1, then the rotation would--as in CORDIC--be accomplished by shifting and adding, giving $X'' = X\cos\alpha_1 + Y\sin\alpha_1$, etc., but now the effect on the length of the

vector will be different in every case and must be computed. This might be accomplished at the end of the process by another ROM which is entered with the set of indices (of the α_1's closest in value to the successive ϕ's). The corresponding contraction factor K given by the following formula could be precomputed and stored in a small table:

$$K^{-1} = \sqrt{1 + 2^{-2(i-2)}}$$

This is to be multiplied by X" and Y", which eliminates two of the four multiplications in the direct rotation computation. For large i, K^{-1} reduces to $1 - 2^{-2i+3}$ --a shift and addition.

The above method for computing the magnitude and phase of a complex number lends itself to pipelining (much as the CORDIC method itself) so that where a string of numbers are to be processed, the processing period corresponds to a single stage of the pipeline.

G. MULTIPORT SERIAL MEMORIES (FFT, CORRELATION)

Serial memories are naturally suited to the processing of serial data streams, which pervade military signal processing. The examples we have discussed include the FFT, image processing, statistical analysis, and linear filtering. Conceptually, the shift register seems perfectly suited for storing and transferring serial data but they consume more power (every bit is transferred on every clock cycle) and occupy more silicon area than random access dynamic memories. There is a growing preference for using sequentially addressed random access memories rather than shift registers. However, there may be some merit to designing memories specifically for serial writing and accessing.

For this purpose, the row and column decoders (which account for numerous layers of logic and a large fraction of the

memory chip area) can be eliminated and replaced with a simple shift register chain for addressing the memory location. The length of the row and column shift registers is the square root of the number of bits in the array. In this way a 1024-bit shift register can be replaced by two 32-bit shift registers and the 1024-bit storage matrix.

A logical schematic is shown in Fig. 9. One of the shift register loops (the row shift register loop, as shown) goes through a complete cycle for each shift of the other shift register loop.

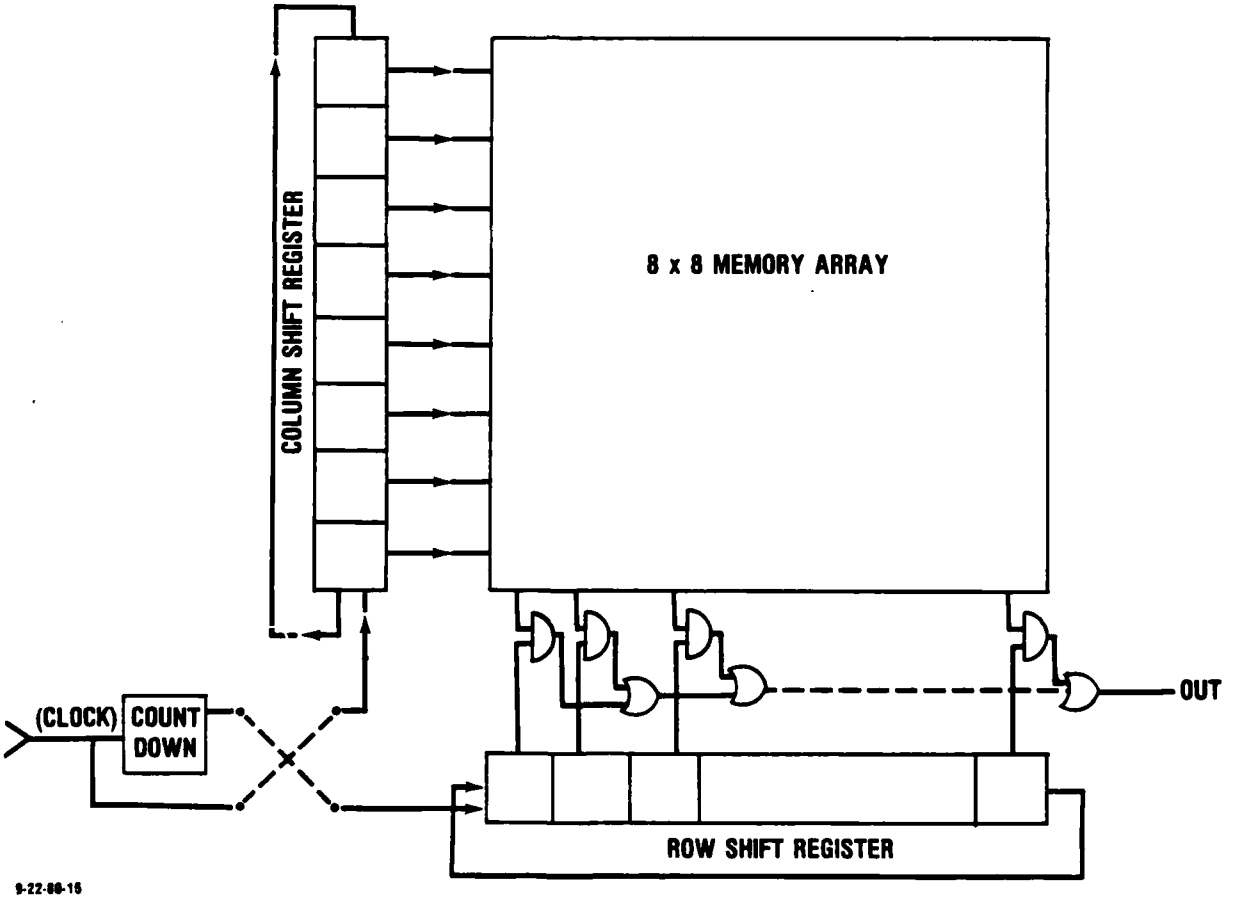


FIGURE 9. Sequentially addressed "corner turning" memory

The degree to which this type of memory would offer any advantage in speed, power consumption, or density would depend on the detailed circuitry in which it is embodied--an issue not addressed here. It should be noted that a complimentary metal oxide semiconductor (CMOS) embodiment of the shift register would greatly reduce the power consumption since only four shift register elements are switched in any cycle (those containing the "1" in each shift register (SR) chain and their succeeding neighbors).

H. MATRIX TRANSPOSE OR "CORNER TURNING"

A memory circuit of this type might find application wherever shift registers are needed (as in the FFT memory circuit of Fig. 6); and in special macros for data sorting or statistical analysis.

The sequentially addressed serial memory, Fig. 8A, requires only a very minor modification for accessing a stored matrix or its transpose in sequence; namely, by switching of either the column or the row shift register (or counter) at the higher rate. For this purpose, the number of rows and columns used must correspond to the number of rows and columns in the actual matrix, or the shift register strings must be programmable as to the effective number of stages.

I. MULTIPORT MEMORIES

The utility of the 4-port memory (two addresses for READ, or WRITE) is illustrated by the self-ordering memory. Suppose a continuous stream of data enters a system and it is necessary to order the most recent N sample according to magnitude. This circuit might then provide the maximum, minimum, median, etc., of the data. This process can be embodied in a network containing three 4-port memories, as will now be shown.

The process of sorting a file has been extensively investigated, but the peculiar difficulty with the problem set forth here is that on each cycle the oldest datum in the file must be located and overwritten by the incoming datum, following which the new datum must find its proper position in the sequence. For this purpose, a counter and three separate files are assumed to exist, together with control circuitry. The counter resets to zero when it reaches the total sample size. The value in the counter will be denoted by A. The current count (A_c) pertains to the oldest datum in the file and also to the newest one by which it is replaced. One of the files will contain rank R filed according to A, another will contain the data D filed according to rank R, and the last will be count A filed according to R.

The procedure according to which the file updates itself is as follows:

(1) With the current count A_c, enter the file R(A) for R_c (the rank of the oldest datum in the file).

(2) Enter the file D(R) with $R_c \pm 1$ giving D_H, D_L. These will be used to determine whether the new datum D_c must move up or down in rank from the position of the datum which it replaces.

(3) If $D_1 \leq D_C \leq D_M$, no further action is required.

(4) If $D_H \leq D_C$, then the new sample must be given a higher rank than R_C and the files R(A) and A(R) must be modified accordingly; this is done by
(a) entering A(R) with $R_C + 1$ giving A'.
(b) interchange the data in R(A) at the addresses A' and A_C, interchange the data in A(R) and D(R) at the addresses R_C and $R_C + 1$. Again, these can be done concurrently.
(c) increment R_C and repeat until $D_C \geq D_H$.

(5) A similar procedure is followed if $D_C < D_L$.

The simultaneous interchange of data called for in step 4(b) requires a 4-port memory circuit.

The number of cycles consumed in taking each datum to its proper position depends on the position of the expiring datum and the ultimate position of the new one, in the extreme case, a total of $\log_2 N$ steps. However, the use of an input first-in-first-out (FIFO) buffer for storing the unranked data until the sorter is ready to receive them will shorten the average processing period somewhat below the worst-case value.

Parallelism can be achieved in batch sorting (where the data are received in a block) by the use of special sorting trees consisting of a series of columns in which pair-wise data comparisons and interchanges are performed.

J. SWITCHING CIRCUITS

Switching circuits are an essential component of processor arrays and of reconfigurable pipelines and should not be overlooked as potentially useful hardware macros. The potential candidates here are large word multiplexers and demultiplexers, register programmable and field programmable cross-bar switch arrays. For "packet" switching purposes an on-board decoding circuitry may be necessary.

Two interesting examples of special switching circuits are the "butterfly" switch (Ref. 17), and the DIMOND switch (Ref. 3). The butterfly switch consists of parallel columns of nodes interconnected by a pattern of lines identical to that of a hardwired FFT. Each signal carries a bit pattern (one bit for each column, or $\log_2 N$ total--N being the number of input and output nodes), the first bit directs the signal ("up" or "down") at the first node, the second at the second node, and so on. The control bit pattern selects the output port and is independent of the port at which the signal originates. The DIMOND modular switch has two input ports and two output ports from which various routing and sorting functions can be synthesized.

Programmable memory chips can be efficiently embodied which function either as a two-port RAM or two-port stack FIFO or first-in-last-out (FILO). An example of such a device is a 16 x 12 multiport RAM/FIFO with an access time of 75 nsec (Ref. 29). The circuit was designed for a high-speed micro signal processor in which it plays a critical role (1) as a stack for instructions between a sequencer and an address generator--which operate asynchronously, (2) as a stack in the input/output (I/O) interface section, and (3) as a RAM in the arithmetic processing section; this particular circuit is in CMOS/SOS, 5μ minimum feature sizes, consists of about 4000 transistors, and uses 64 pins.

One of the more important uses of more refined design rules (smaller features) is larger on-board two-port RAMs for working storage (and index systems), and stacks for sequencing and buffering between asynchronous processing elements.

K. PROGRAMMABLE MACROS

Several attempts have been made to analyze military systems' processing requirements to assess the applicability of various monolithic macros (Ref. 8, for example), which tend to confirm the impression that hardware macros, as a group, do not have a commonality of application remotely resembling that of the standard medium-scale integration (MSI) circuits. This motivates some design groups to seek programmable macros, of which the so-called MATH chips are an example (Ref. 30), although these may too nearly resemble fully programmable processors to serve as a good example.

Among the hardware macros listed above, the following groups are candidates for embodiment as monolithic programmable hardware macros:

(1) Complex multiply, magnitude, correlation, FFT butterfly, add/subtract;

(2) Peak, median, sort and merge, histogram, data reordering, matrix transpose;

(3) Sine/cosine, logarithm, exponential generator;

(4) Floating point conversion, barrel shifter.

The FFT full butterfly takes two complex numbers (Z_1 and Z_2, say) and a coefficient of the form $e^{i\phi}$ and forms $Z_1 + Z_2e^{i\phi}$ and $Z_1 - Z_2e^{i\phi}$, involving one complex multiplication (which in itself consists of four real multiplicatcns and two real additions, with signs) and two complex additions; a total of four multiplications and six additions with signs. A monolithic FFT butterfly would contain the resources for all the other functions in group (1), i.e., complex multiplication, magnitude, correlation, and add/subtract.

All of the computations and data processing in the second group requires storing and comparing of a group of data. The median and histogram necessitates numerical ordering of the data.

The third group of functions of a single variable are calculable using the CORDIC algorithm or might be generated by table look-up with interpolation or extrapolation.

In the fourth group, the barrel shifter or normalizing circuit is an integral part of floating point conversion.

The programmable macro would be somewhat more complex than a fixed macro containing, as it must, control structure, but this would increase its commonality.

L. PROGRAMMABLE SIGNAL PROCESSING

The IBM Advanced Signal Processor (ASP) (Fig. 10) typifies an architecture for programmable general-purpose signal processors

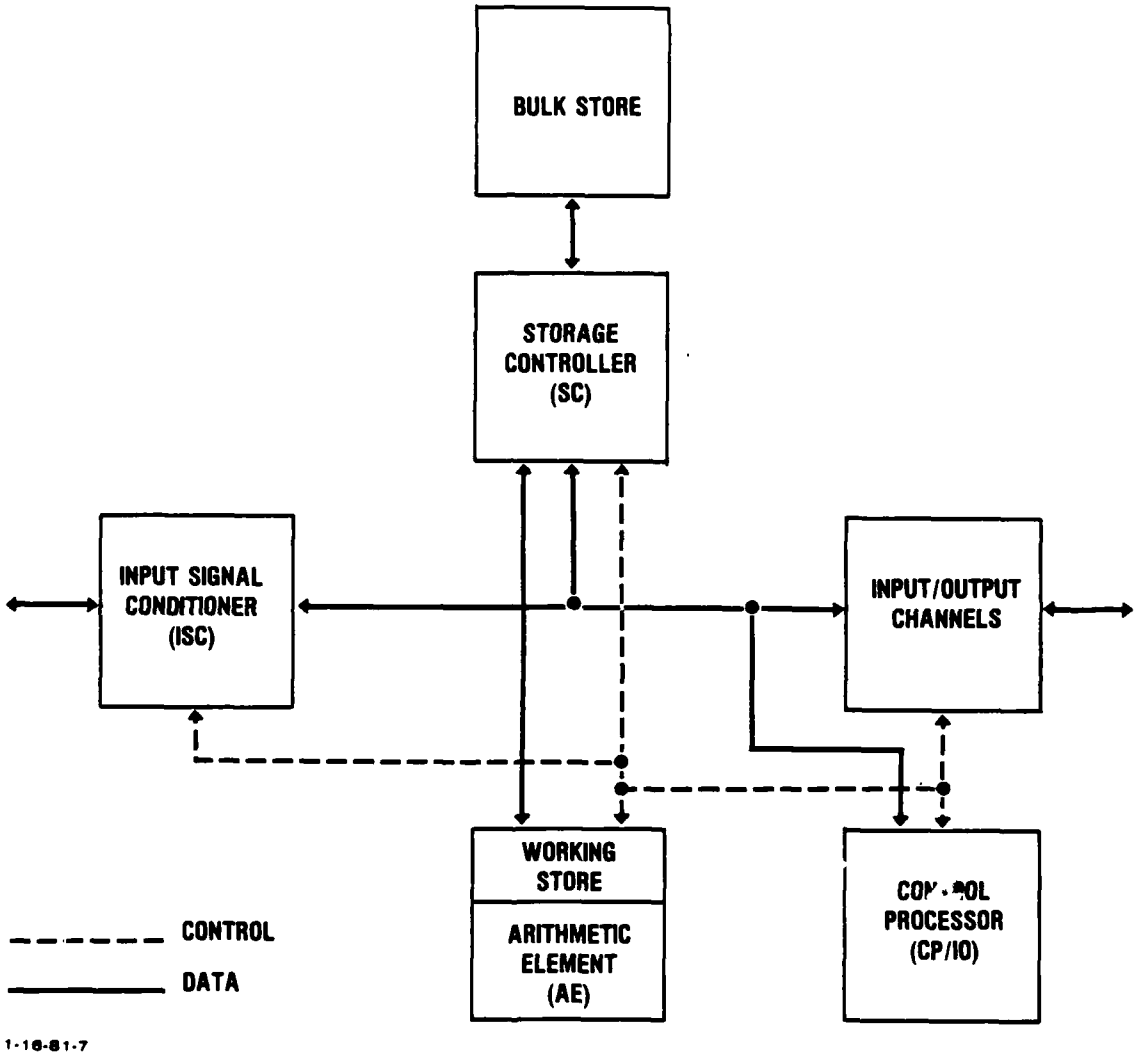


FIGURE 10. IBM advanced signal processor analyzer unit organization (courtesy IBM)

on which many systems are based. Basically, it consists of three processing elements operating concurrently; a memory management unit [the signal controller (SC) in ASP, the input-output processor (IOP) in the APS-41, etc.], an arithmetic unit (which may consist of several pipelined processors acting in parallel), and a controller unit which coordinates the operation of the other units and generally executes a comprehensive instruction set for the operations associated with post detection (peak picking, thresholding, coordinate changes, etc.).

However, many other, apparently opposing, signal processing architectural approaches have also been identified. Large-scale computational power, in terms of instruction set, word length, implementation technology, capacity, etc., is represented by the Lawrence Livermore S-1 Mark IIa processor. An inherent drawback of this approach for embedded application appears to be a lack of modularity--the ability to incrementally add to a system's computational power in a cost-effective manner. At the other extreme, the Texas Instruments Micro Vector Processor (a signal processor of modest computational power) is designed to be configured in distributed processing systems in a functionally dedicated manner. Commercial architectures, e.g., Honeywell HAP, Raytheon Co. ADSP, and the IBM SD-300, fall into one or the other of the two classes--Power Centralized or Functionally Modular, Distributed. These approaches need to be examined relative to VHSIC implications: the number of chip types, including identification of macrocell functions required, remains to be determined for both architectural approaches. A third approach, the functionally partitioned semi-programmable processors (already alluded to), has apparently not yet been implemented.

A system, such as S-1 Mark IIa, designed without regard to level of circuit integration, really does not address the physical embodiment requirements of military systems on which

the VHSIC program focuces. The functionally partitioned semi-programmable (or parameter programmable) processors utilize the data flow structure of the non-data-dependent functions [i.e., finite impulse response (FIR) filter] to physically link multipliers, adders, etc., to realize these functions. Processing is performed "on the fly" without a main memory. Basic elements of a possible chip include programmable shift registers, multipliers, adders, etc., and others discussed above. The programming of such a chip would be through the connectivity of the shift register elements to one another and to the operation of the multipliers and adders. This could be fuse programmable or control-word programmable.

The functionally partitioned "parameter programmable" processor would be most appropriate for the arithmetically intensive high throughput applications, such as adaptive processing, digital filtering, fast Fourier transform (FFT), etc.

As a rule, these stages are followed by a detection or decoding process which substantially reduces data rate, but the remaining data requires more varied treatment (such as ambiguity resolution, track association, target signature identification) which calls for a much larger instruction set and is not so suitable for pipelining. Finally, the data which survives is passed on to the displays, control units, communication encoders, etc.

These are some of the considerations which led to the concept of a processing system architecture, shown in Fig. 2 (Ref. 31). The signal-conditioning section is largely linear, but contains the analog-to-digital (A/D) converter and may contain a partially digital automatic gain control (AGC) function for floating point conversion, etc. The parameter programmable section would consist mostly of hardware macros, programmable interconnections, and would utilize circuits of the highest

functional throughput capacity--which are the principal focus of the VHSIC program. If this concept proved generally valid, the PSP portion might be reducible to a monolithic circuit under VHSIC. Several monolithic-programmable general-purpose processors are under study.

The full range of data transfer operations (testing, shuffling) must be dealt with in the semi-programmable elements which would, in most cases, require a separate storage control processor.

M. CONTROLLER

In many applications, the arithmetic and data transfer operations can keep pace with incoming data streams only when carried out concurrently, such as by separate processors dedicated for each purpose and operating in parallel. The coordination of these concurrent processes is often performed by a third unit, the controller, or host computer, which in addition to supervising and coordinating the operation of the storage control (or IOP) and arithmetic units (AU) may execute a comprehensive instruction set for general-purpose computer applications (Refs. 13, 26, 32).

The arithmetic unit executes optimized algorithms and macroinstructions (such as complex multiplication and addition, normalization), and its software tends to be portable. The software for the controller, on the other hand, is particularized to the system in which it is embedded and to the computational resources of the storage control and arithmetic units it supervises. This software often proves to be expensive to generate and maintain, and is rarely portable. However, the software burden would be considerably alleviated by the development of an efficient signal processor higher order language (HOL) and a problem-oriented language (POL) to generate the HOL control statements to link and sequence macros. Again, some of the work

on Ada compilers appears promising, although Ada itself has been criticized for insufficient parallelism and concurrency.

The host computer or controller can also provide important services in program checkout and debugging by being able to stop the processor at any time, inspect its memory (status, command, data) and registers, and reinitialize its operations.

N. ARITHMETIC PROCESSOR

Signal processing arithmetic units typically contain pipelined multipliers, adders, scalers, sine/cosine look-up, logic units, etc., often in arrays operating in parallel from a common instruction sequence. Such units are microprogrammed by microinstructions of 50 to 75 bits in length. The micro-instruction ROM typically contains several thousand instructions with access rates of 100 μsec or less.

In some cases, the arithmetic until will consist of six stages of pipelining with new inputs and outputs every 100 μsec or less.

The more powerful processors can perform two complex adds, one complex multiply, a trig look-up, and a variety of data transfers, and can perform an FFT butterfly or a 2-pole-2-zero filter stage in a single instruction; scaling (left or right shifts) of all inputs independently is often provided for. As a rule, the arithmetic unit contains its own microinstruction decoder and control generator.

An especially important processing function in the arithmetic processing section of a programmable signal processor is scaling the variables (Ref. 29). This prevents saturation and needless loss of significance, it is an integral part of floating point conversion, and permits important efficiencies in the size of stored tables.

The separate operations involved are:

(1) Storing and ordering data

(2) Counting the number of lead zeros in the largest number (priority encoder)

(3) Determining the desired shift (scale register logic)

(4) Generating the shift count either from scale register logic or externally supplied scale factors.

Such a circuit is part of the Raytheon High Speed Micro Signal Processor, the TCS 143 (CMOS/SOS*, 5μ minimum feature sizes, 4,800 device, 64-pin package).

The likely contributions of VHSIC to the arithmetic processor include the monolithic complex multiplier, the macros, such as adder/subtracter, two-port stacks, 16- and 32-bit arithmetic-logic units, and a barrel shifter.

0. ADDRESS GENERATION AND KERNEL COMPUTATION

The implementation of address generation varies considerably among different SP designs. Since DO loops are characteristically required for the execution of signal processing algorithms, a bank of internal address registers is often included, any of which can be decremented and tested for zero with a branch on zero. Alternatively, the address registers may be incremented or decremented by powers of 2 or by an amount stored in special increment registers. This provides a means for table look-up and indexed memory accesses.

*Complimentary metal oxide semiconductors/silicon on sapphire

These features give the appearance of being ad hoc, but actually the generation of addresses in the execution of SP algorithms can be analyzed with some generality, showing the utility of various specific computational resources (Ref. 8). Most of the arithmetically intensive operations in military signal processing consist of nested DO loops with a kernel operation consisting of vector products. Included in this class of algorithms are the FFT, beam forming, adaptive processing, linear filtering, coordinate transformation, Kalman filtering, image processing, etc.

A simple case in point is the multiplication of a matrix by a vector, which occurs in Kalman filtering, image processing, coordinate transformation, adapti· processing, etc. The corresponding program would be:

```
FOR I = 0 UNTIL M - 1 DO
FOR J = 0 UNTIL N - 1 DO
 Y(I) = Y(J) + C(I,J)*X(J)
END
```

Computer memories are implemented in one dimension, which means that the elements of the matrix C (I,J) are usually stored in rows or columns so that the address of (I,J) elements equals start address plus (I - 1) * N plus (J - 1). This is equivalent to forming the address (by catenating I,J), K - I * 2^N + J. The indicated addition is not actually executed because there is never a carry.

Stored in this sequence, the matrix elements C (I,J) are addressed by incrementing the address register by one at each step of the calculation. This requires very minimal circuitry. However, if the transpose of the matrix also appeared in a similar calculation, the addressing would involve incrementing the address register by M, with an additional increment of one

after N steps. Random access to the elements of C would involve both addition and multiplication. For this reason the S-1 (Lawrence Livermore) SP contains a matrix transpose macro which is said to be cost effective. Such a macro would appear to be best suited for adaptive processing which, in its most general form, involves matrix inversion. A sequential circuit for accessing a matrix or its transpose has already been described.

P. SIGNAL CONDITIONING, DYNAMIC RANGE, AND DATA FORMAT

The distinctive character of signal processing is nowhere more evident than in the appropriate word size--the number of bits carried. This is related to the dynamic range of the front end linear amplifier (fed by the sensor) and by the signal dynamic range--the power ratio of the strongest signal (which may be ground clutter echoes in radar, electronic countermeasures in communications, or surface ship emanations in sonar) to the weakest signal of interest (moving target echoes, submarine emanations).

The dynamic range for radar and communication receivers using the best current technology is 70 dB to 80 dB or so which would permit representation of both the peak signal and of receiver noise with 12 to 15 bits.* This is the instantaneous or receiver dynamic range. Any precision in excess of this is apparently superfluous and wasteful, provided the receiver gain preceding the A/D converter keeps the signal voltage within the range of the A/D converter. This is accomplished by the automatic gain control (AGC) which acts as a variable multiplier of the signal voltage. In a floating point representation this multiplier corresponds to the exponent while the digitized signal

*Allowing 1 bit per 6 dB of dynamic range. The dynamic range (the ratio of the saturation signal power to the receiver noise power) corresponds to $\log_{10}\{2^{2B}\}$ when the least significant bit is set at the noise level.

corresponds to the mantissa. Ideally, the AGC would prevent saturation of the receiver on the strongest signals while preventing the weakest detectable signals from falling below the noise level of the front-end (signal conditioning) section. These conditions can only be approximated in a real signal environment, in part because the time constant of the AGC circuit itself cannot be made to correspond to fluctuations in the strong interfering signals (ECM, clutter echo, interference) without introducing modulation distortion into the desired signals.

In passing, it should be remembered that the signal word size which must be carried ultimately depends very strongly on the amount of digital signal integration and the spectral composition of the sought and interfering signals. If the interference is "white" noise-like and the integrated gain exceeds the input ratio of interference to the signal, then the front end may be hard limited (1 bit of dynamic range, no AGC) with an inconsequential loss in detection sensitivity. This condition occurs only in exceptional cases, although it may be seen more frequently in future systems, particularly in satellite platforms and in low probability of intercept (LPI) radars and communications systems where the highest signal integration is sought.

The antenna structure itself plays a corresponding role since the formation of a beam from a large array of elements is nothing other than signal integration where the independent data are spatial rather than temporal samples. Thus, in major sonar systems where the combination of beam forming and spectral analysis produces an integration gain of some 60 dB, the initial signals at the receiving elements are hard limited, preserving only their polarity--another instance of the great variability among the various signal processing applications. However, when the angular distribution of the noise field is not isotropic--corresponding to "colored" noise, i.e., a non-uniform

power spectrum--the loss in detection sensitivity from limiting can be considerable.

The relationship between dynamic range and word size follows from a straightforward analysis. The number of bits b in the fraction (the direct receiver channel) must be

$$b = 1 + \frac{r\log_2 10}{20}$$

for a receiver dynamic range r (in dB), while the number of bits B in the exponent for a total signal dynamic range R is

$$B = \log_2 \left[\frac{R\log_2 10}{20}\right].$$

Thus, for r = 70 dB, R = 150 dB

b = 12, B = 5.

These appear to be generously large figures and stand out in contrast to commercial data processing uses where the figures b = 24 and B = 8 are often suggested as minimum standards and b = 60, B = 11 as maximum. The F-18 radar signal processor and the Multimode Radar Processor are examples of fixed-point 12-bit systems.

The above refers to the word size at the output of the signal conditioner. Further along the signal processing chain, larger words may be required to prevent algorithmic aberrations such as limit cycle oscillations in feedback loops.

Q. DATA TRANSFERS

The constant flow of signals into the SP presents an often difficult problem of data management, except in full hardwired embodiments or distributed processing systems in which the

processing elements keep pace with the data flow so that data buffering is not needed. When data must be assembled from several sources or where blocks of input data are collected and then analyzed (as in the FFT) or where the tasks other than processing input data share the processing resources, then, in these and other cases, the input data must be stored in bulk memory and subsequently transferred to working storage for processing.

The complexity of the data management hardware depends on a number of factors, such as the incoming data rate relative to the memory cycle time and the quantity of data which is processed in a block, etc. If the incoming data rate exceeded the memory bandwidth, the data would have to be distributed (demultiplexed) among parallel memory elements, although this is rarely the case in current practice, simply because A/D converters of 12 bits or more precision do not now operate at speeds much in excess of bipolar memories. High speed data arriving in bursts can (as in ESM applications) be loaded into memory through a high speed FIFO buffer. Such a circuit is a candidate for development under the VHSIC program. The alternative to buffering is a two-port memory or dual memory, one section of which is alternately read out while the other is receiving data.

The data must then be transferred into the working storage of the processor itself where the necessity for speed (in further transfers to the input registers) places a limit on the available memory space. Here again, parallel or two-port memories are the only means to avoid interrupting the processing for data transfers to and from bulk memory.

These difficulties must be dealt with both in the design (organizational architecture) and in the software. At the design level, separate dedicated components of the processor (storage controllers) manage data transfers (sometimes through

several levels of memory) which are equipped with sequencing circuitry--a more or less elaborate section having some arithmetic elements for precomputing addresses for data transfers. In the IBM advanced signal processor, the storage controller (SC), like the arithmetic elements, operates under the direction of a central processing element. The SC is the appropriate apparatus for doing error detection and correction on stored data before passing it on to the arithmetic processing unit. These features of the SC stand in contrast to the simple processors of yore where data addresses were precomputed and stored or computed at running time as an activity of the central processing unit using its arithmetic processing resources (memory fetch instructions sometimes consisted of a dozen or more sequential operations). The signal processor must have special resources dedicated to data management, but these special resources require additional coding, placing an added burden on the programmer.

The programmer's difficulties would be considerably relieved by an adequately powerful HOL compiler--powerful in the particular sense of generating efficient codes for data management. Unfortunately, little has been done toward the development of such a compiler, even for processors with simple data management resources. However, the importance and dimensions of the problem are becoming better understood and a compiler for Ada now being developed at Carnegie Mellon University is expected to produce efficient data management, taking into account the sequencing and other data management resources of any particular SP.

In some commercial systems the storage control functions are taken over by an IOP which may be multiplexed for handling several asynchronous data channels using flexible address and control sequencing. The IOP may be capable of servicing several programs simultaneously on a priority basis according to the demands and availability of peripheral devices.

R. SOFTWARE SUPPORT

Although the subject of this study is circuitry and not software support, per se, the implications of the quality of software support cannot be ignored. Past experience has shown, in fact, that the successful use of a SP in a system often depends as much on the software as on the circuitry. However, the question of software support must also be addressed for the compelling reason that at higher levels of integration the choice of circuitry comprises a commitment to specific software features as well as an opportunity to remedy past deficiencies in software support.

On the hardware side, the circuitry which relates most closely to software includes the hardware macro, the memory control circuit and sequencer (with their own computational resources), and the number and size of microcode and working storage registers.

On the software side, new and more powerful compilers are needed that will be capable of utilizing the computational resources which will become feasible under VHSIC. Among the burdens which must be taken on by the compiler is the efficient use of special circuitry for the transfer, validation, and reordering of data. It should be accepted as a goal that transfers in and out of the arithmetic working storage be made transparent to the user.

At an even more fundamental level, the development of a set of VLSI circuitry which could be effectively and economically integrated into a variety of future military systems can proceed with confidence to the extent that a comprehensive hardware description language and also hardware design language can be put in place which will enable the necessary communication among the system designer, the analyst, the circuit functional designer, the circuit processor, the end user, and the operation support organization.

REFERENCES

1. Glenn W. Preston, *Large Scale Integrated Circuits for Military Applications*, Institute for Defense Analyses, IDA Paper P-1244, May 1977.
2. Glenn W. Preston, *High Speed Integrated Circuits for Military Applications*, Institute for Defense Analyses, IDA Paper P-1423, November 1979.
3. Pierre G. Jansen and J.L.W. Kessels, "The DIMOND: A Component for the Modular Construction of Switching Networks," *IEEE Trans. Comp. C-29*, p. 884, October 1980.
4. Steven J. Kartashev, et al., "Supersystems for the 80s," *Computers*, Vol. 13, No. *11*, November 1980.
5. Leonard B. Weisberg and Larry W. Sumney, "The New DoD Program on Very High Speed Integrated Circuits (VHSIC)," *GOMAC 1978 Digest of Papers*, p. 18.
6. AGED Special Technical Area Review on Design Automation, Institute for Defense Analyses, August 15, 1980, GED-L 80/4, prepared by Palisades Institute for Research Services, Inc.
7. "Issues Relating to Signal Processing Standardization," Ad Hoc Study Summary, Signal Processing and Display Committee, NSIA Study Team Draft, 12 December 1979.
8. James W. Bowra, *Multisensor Standard Macro Function Study*, Honeywell for Naval Air Development Center, Warminster, PA, 1 December 1979.
9. Glenn W. Preston, "The Hardware Macro as a Means of Raising the Effective Figure of Merit of Integrated Circuits," *GOMAC '80 Digest*/Papers, p. 186, November 1980.
10. Jonathan Allen, "Computer Architecture for Signal Processing," *Proc. IEEE 63*, No. *4*, April 1975, p. 624.
11. Glenn W. Preston, *Register Programmed General Logic Circuits*, Institute for Defense Analyses, IDA Paper P-1336, June 1978.
12. The BBN Pluribus, Private correspondence.
13. Kenneth J. Thurber, *Large Scale Computer Architecture*, Hayden Book Co., 1976.

14. D.H. Daggett, "Decimal-Binary Conversion in CORDIC," *IRE Trans Elect. Comp.*, Sept. 1959, p. 335.

15. S.C. Pohlig and J.M. Frankovich, "Efficient Multiplier for the FFT," MIT Lincoln Laboratory (undated internal memorandum).

16. Earl E. Swartzlander, "Merged Arithmetic," *IEEE Trans. Comp. C-29*, p. 946, October 1980.

17. R. Rettberg, et al., *Development of a Voice Funnel System: Design Report*, Bolt, Beranek & Newman, Inc., Report No. 4098, August 1979.

18. *HL Cell Library*, Harris Semiconductor, November 1977.

19. E.G. Fubini and Smith, A.S., "Limitations in Solid-State Technology," *IEEE Spectrum*, p. 55-60, May 1967.

20. George E. Avery, "An Imperfect Image? VLSI's Future in the VHSIC Mirror," *Military Electronics/Countermeasures*, August 1980.

21. Ben Gold and T. Bially, "Parallelism in FFT Hardware," *IEEE Trans. Audio Electroacoust. AU-21*, pp. 5-16, February 1973.

22. H.L. Grogansky and G.A. Works, "Pipeline Fast Fourier Transform," *IEEE Trans. Comp. C-19*, p. 1015, November 1970.

23. Leo W. Lemley, "VHSIC Applications in EW/ESM," 1980 DOD/AOC Technical Symposium, El Toro Marine Air Station, California, October 1980.

24. Stephen P. Mosier, "The AN/GSX-2: A Real-Time Signal Processor for Intelligence Support," 1980 DOD/ACC Technical Symposium.

25. R.M. Lougheed, D.L. McCubbrey, S.R. Sternberg, "Cytocomputers: Architectures for Parallel Image Processing," *IEEE Workshop on Picture Data Description and Management*, August 1980.

26. Abraham Peled and Bede Liu, "A New Hardware Realization of Digital Filters," *IEEE Trans. Acoust. Speech Signal Processing, ASSP-22*, pp. 456-462, December 1974.

27. Jack Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. Elect. Comp.*, p. 330, September 1959.

28. U.S. Patent 3,766,370, "Elementary Floating Point CORDIC Function Processor and Shifter," October 1976.

29. *High Speed Micro Signal Processor Interim Report*, Raytheon Co., BR-11063-1, 2, March 1979.

30. "Now for the Math-Processing Chips," *Electronics*, p. 99, July 5, 1979.

31. Harley A. Cloud, IBM Federal Systems Division, Manassas, Virginia, private correspondence.

32. *General Characteristics of the SPS-41 and SPS-81 Programmable Digital Signal Processors*, Signal Processing Systems, Inc., Waltham, Massachusetts, undated memorandum.

33. Glenn W. Preston, *Integrated Circuit Characteristics for Future Military Avionics*, Institute for Defense Analyses, IDA P-1433, January 1980.

APPENDIX

MILITARY SIGNAL-PROCESSING REQUIREMENTS

APPENDIX

MILITARY SIGNAL-PROCESSING REQUIREMENTS

The military applications of integrated circuits (ICs) which demand the highest functional throughput rates are the subject of two earlier studies (Refs. A-1, A-2).

The actual sources of high-speed signal processing include:

(1) The FFT algorithm applied to acoustic data for beamforming and signature analysis and to radar data for pulse compression and synthetic aperture formation and numerous other applications;

(2) Matrix inversion, the computation of the covariance matrix and its inverse in adaptive processing for antenna array formation and filtering operations such as moving target indication (MTI) in radar;

(3) Digital filtering: infinite impulse response (IIR) and finite impulse response (FIR);

(4) Coordinate transformation;

(5) IR and optical image processing algorithms (edge detection, shape recognition, spatial filtering, gradient projection, rotation, distortion compensation, and so on);

(6) Bandshifting; in radar, communication, electronic surveillance measures (ESM);

(7) Track association, track extrapolation, and smoothing;

(8) Correlation in signal decoding, passive triangulation;

(9) Computation of magnitude; in radar, sonar, image processing, communications;

(10) Peak detection and thresholding.

Tables A-1 and A-2 indicate the widespread utilization of these operations.

A. FAST FOURIER TRANSFORM

The fast Fourier transform (FFT) is one of the most important special signal processing algorithms found in military systems. Rather large FFT (512 points or more) are used in acoustic beam forming and in real time (electronic) synthetic aperture array processing. Actually, in acoustic applications one FFT is used for beam forming and a separate FFT is performed on each beam for spectral analysis. Similarly, in some forms of synthetic aperture radar one FFT performs a function analogous to pulse compression in which a linearly FM pulse is converted into fixed tones (pitch here corresponding to range, not doppler shift) then a second sweep-to-sweep FFT is used to form the focus synthetic aperture.

In the sonar example, the instruction rate (multiplications and additions per second)

$$\dot{C} = \frac{4W\Omega}{\beta} \log_2 \frac{\Omega}{\beta} + \log_2 \frac{W}{\delta f} ,$$

when W is the total signal bandwidth, δf the final spectral resolution of the FFT, Ω is the total (solid) angular sector being monitored and β the final (solid) angular resolution. Typically, in current systems, $W = 500\sim$, $\frac{W}{\delta f} \sim \frac{\Omega}{\beta} \sim 10^3$ and

$$\dot{C} = 40 \text{ MIPS} .$$

TABLE A-1. VHSIC SIGNAL/DATA PROCESSING FUNCTIONS

VHSIC System Candidate	Target Recognition	Spread Spectrum Waveform Processing	Adaptive Beam Forming	Multisensor Correlation	Moving Target Doppler Processing	High Resolution Ground Mapping	Signal Integration	Sorting & Parameter Analysis	Moving Target Tracking	Image Processing	Coding & Decoding
Multimode Fire and Forget Missile	X				X	X	X	X	X	X	
Battlefield Info Distribution		X		X			X				
High Mobility Integration EW Weapons Targeting	X	X		X	X			X	X		X
Advanced Target Acquisition/FCS	X	X		X	X		X	X	X	X	
Programmable Acoustic Signal Processor	X		X	X	X		X	X	X		
Programmable AJ Communications Modem		X					X	X			X
Advanced Signal Processing for Airborne and Shipboard Surveillance Radars	X	X	X	X	X	X	X	X	X	X	X
Tactical Aircraft Programmable Radar Signal Processor	X	X		X	X	X	X	X	X		
ESM Signal Sorter	X			X	X			X	X		
Digital Processors For Imaging Systems							X	X	X	X	
General-Purpose Computer	X			X				X	X		X
Subsystems for NATO Identification Systems	X	X	X		X			X	X		X
Multifunction Radar Signal Processor	X	X	X	X	X	X	X	X	X		
VHSIC Programmable Comm. Signal Processor for JTIDS		X	X				X	X			X
Shrink of a General Purpose Processor	X			X				X	X		X
Autonomous Cruise Missile Guidance	X				X	X	X		X	X	
Advanced Power Management for EW Applications	X	X		X	X		X	X	X		X
Advanced Onboard Signal Processor (AOSP)	X	X		X	X	X	X	X	X	X	
Advanced Medium Range Air-to-Air Missile (AMRAAM)				X	X			X			X
VHSIC Universal Sensor Signal Processor for E-3A	X		X		X		X	X	X		

TABLE A-2. TYPICAL ALGORITHM UTILIZATION IN VARIOUS SYSTEMS TYPES

Algorithm (Macro-Operation) \ System Area	Surveillance	C^3	ASW	Weaponry	EW/ESM
FFT, 1-D	X		X		
FFT, 2-D	X			X	
Digital Filter	X		X	X	X
Correlation, 1-D	X		X		
Correlation, 2-D	X			X	
Vector Multiplication	X	X	X		X
Integration	X	X	X	X	X
Normalization	X		X		X
Thresholding	X	X	X	X	X
Magnitude	X	X	X	X	
Trigonometric Function	X	X	X	X	X
Logarithmic Function	X	X	X		X
Matrix Inversion	X	X	X		X
Find Max/Min	X	X	X	X	X
Peak Pick			X		
Sort	X	X	X	X	X
Compare	X	X		X	X

The corresponding relation for synthetic aperture radar

$$\dot{C} = \frac{2\Delta RV}{\rho^2} \log_2 \left(\frac{R\beta}{\rho}\right),$$

in which ΔR is the width of the surveillance swath, V the platform velocity, β the beamwidth of the physical aperture and ρ the resolution (both transverse and radial). For $\rho = 0.2$m, $\Delta R = 10^4$m, $V = 10^2$m/sec, $\beta = 3 \times 10^{-2}$ radians, $R = 10^5$m.

$$\dot{C} = 700 \text{ MIPS}.$$

For simple spectral analysis (one level of FFT):

$$\dot{C} = 4\ W \log_2 \left(\frac{W}{\delta f}\right) \ ,$$

in which W is the total bandwidth and δf the final spectral resolution. For ELINT applications, the largest feasible signal bandwidth is covered. If, for example, W = 10 MHz and δf = 1 kHz, $\dot{C}$ = 520 MIPS.

Acoustic example:

$$W = 300 \text{ Hz}, \ \delta f = \frac{1}{8} \text{ Hz} = 3 \times 10^{4}$$

$$10 \text{ beams} \quad \frac{\Omega}{\beta} = 10$$

$$\dot{C} = 1.2 \times 10^{4} \ (\log_2 10 + \log_2 2400)$$

$$= 2 \times 10^{5} \ .$$

B. ADAPTIVE PROCESSING

Many adaptive techniques are used in radar and communication systems to reduce the effect of sidelobe jamming and mutual interference. In the most general method, the gain and phase response of each receiving element is optimized to produce the best ratio of main lobe response relative to all received signal power. The algorithm corresponds to the straightforward solution of a minimization equation. It involves forming the covariance matrix for all of the array elements; N^2 products in all taken at the Nyquist sampling rate followed by summation which gives a sample covariance matrix. The covariance matrix must then be inverted (which involves $\sim N^3$ multiplications) followed by a matrix multiplication (N^2 multiplications) and a dot product (N multiplications), but the latter operations are performed not at the Nyquist sampling rate but at a much lower rate corresponding to the summation period (in the estimation of the

covariance matrix). Even for modest size arrays the arithmetic rate is considerable.

For theoretical purposes it is assumed that the background noise and interference is Gaussian and stationary, and the choice of integration period (in forming the estimate of the covariance matrix) is somewhat empirical. Actually, much less arithmetic need be performed than the above discussion indicates, by embodying the Gram-Schmidt method in which a new set of statistically independent variables are formed (from the signals from each element of the array), so that the covariance matrix is diagonal. This effectively eliminates the matrix inversion step. The corresponding block diagram is shown in Fig. A-1. However, the Gram-Schmidt method has been criticized for these applications.

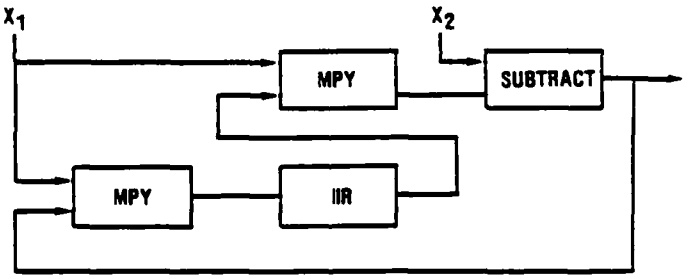


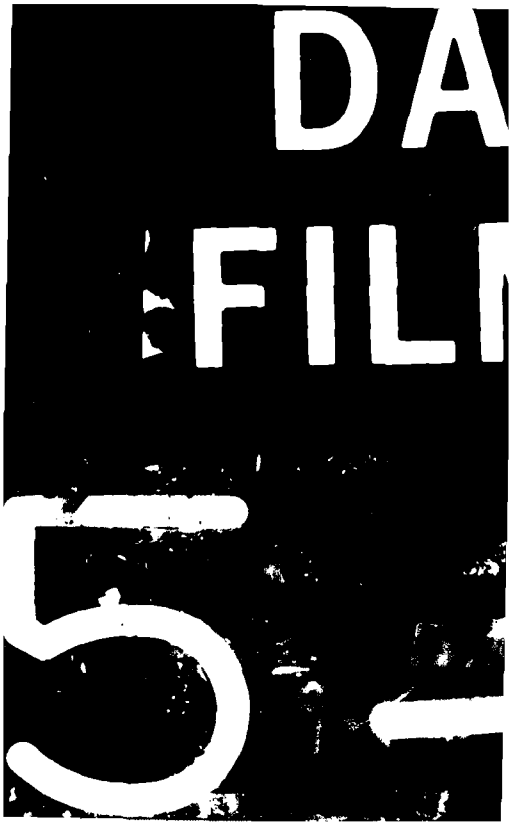
FIGURE A-1. Adaptive array element using Gram-Schmidt technique

There are two independent sources of error in the adaptive procedure, fluctuation in the sample covariance and loss of data in the subtractions which occur in orthogonalization or matrix inversion.

REFERENCES, APPENDIX

A-1. Glenn W. Preston, "The Hardware Macro as a Means of Raising the Effective Figure of Merit of Integrated Circuits," *GOMAC '80 Digest*/Papers, p. 186, November 1980.

A-2. Glenn W. Preston, *High Speed Integrated Circuits for Military Applications*, Institute for Defense Analyses, IDA Paper P-1423, November 1979.


DA
FIL